UNIVERSITÀ DEGLI STUDI DI PADOVA

Facoltà di Ingegneria Gestionale

DIPARTIMENTO DI TECNICA E GESTIONE DEI SISTEMI INDUSTRIALI

Tesi di Laurea Triennale

RISOLUZIONE TRAMITE VISUAL BASIC DEL PROBLEMA DEL TRASPORTO DELLE PELLI FRESCHE DAI MACELLI ALLA CONCERIA

Relatore
Prof. Giorgio Romanin Jacur

Laureanda
Tatiana Rognin

Matricola
563024

ANNO ACCADEMICO 2011-2012

INDICE

INTRODUZIONE

Negli ultimi decenni si `e verificato un crescente utilizzo di pacchetti software basati su tecniche di ricerca operativa e programmazione matematica per la gestione efficiente della fornitura di beni e servizi nei sistemi di distribuzione.

Il grande numero di applicazioni reali, sia nel Nord America sia in Europa, ha ampiamente dimostrato che l'utilizzo di software per la pianificazione dei processi di distribuzione produce un sostanziale risparmio (generalmente dal 5% al 20%) nei costi globali di trasporto. `E' facile osservare come l'impatto di questo risparmio sul sistema economico di un Paese sia significativo dal momento che i processi di trasporto riguardano tutte le fasi della produzione dei beni ed i costi relativi rappresentano una componente rilevante (generalmente dal 10% al 20%) del costo finale.

Il successo nell'utilizzo di tecniche di ricerca operativa è dovuto non solo allo sviluppo hardware e software nel campo dell'informatica e alla crescente integrazione dei sistemi informativi nel processo produttivo ed in quello commerciale ma soprattutto allo sviluppo di nuovi modelli che cercano di prendere in considerazione tutte le caratteristiche dei problemi reali ed alla concezione di nuovi algoritmi che permettono di trovare buone soluzioni in tempi di calcolo accettabili.

In questo lavoro si vuole specificatamente approfondire il Vehicle Routing Problem, strumento principale per modellare la realtà della logistica e del trasporto dei beni, attraverso l'utilizzo di Software come Visual Basic che lo risolvono.

CAPITOLO 1

Problemi decisionali nei trasporti: Vehicle Routing Problems

1.1 Vehicle Routing Problems

Il Vehicle Routing Problem (VRP) `e un tipico problema operativo nelle reti di distribuzione, e consiste nello stabilire i percorsi di una serie di veicoli per servire un insieme di clienti.

Dato un insieme di veicoli con determinate caratteristiche che devono visitare un insieme di clienti (anche essi con determinate caratteristiche) distribuiti all'interno di una rete di trasporto a partire da uno (o più) depositi centrali.

Possibili applicazioni del VRP sono frequenti in problemi logistici e distributivi di dettaglio.

Per esempio, si può pensare a contesti applicativi in cui si debba distribuire al dettaglio (oppure provvedere alla raccolta) un certo bene. In applicazioni reali sono presenti frequentemente molti vincoli aggiuntivi che possono complicare la struttura del problema.

Generalità

Un problema di instradamento dei veicoli può essere definito andando a descrivere nel dettaglio le caratteristiche dei veicoli, dei clienti e della rete di trasporto che ne definiscono il contesto operativo.

Si possono distinguere due grandi classi di problemi, a seconda di come siano configurati i clienti.

Se si assume che i clienti siano uniformente distribuiti lungo le connessioni della rete di trasporto (ad esempio lungo una strada) allora si parla di problemi di arc routing, mentre se i clienti sono rappresentabili come entità distinte (ad esempio nodi di un grafo) si parla allora di problemi di node routing.

Tra i problemi di node routing si possono definire vari casi a seconda delle differenti modalità operative del contesto applicativo preso in esame.

Una prima distinzione si può fare andando a considerare, ad esempio, la capacità di carico di un veicolo.

Se questa risulta superiore alla somma delle domande di tutti i clienti allora il problema del VRP si semplifica e si configura come un problema di Travelling Salesman Problem (TSP), altrimenti,nel caso in cui la somma delle richieste di tutti i clienti risulti superiore alla capacità di trasporto di un singolo veicolo si parla allora propriamente di problemi di Capacitated Veichle Routing(CVRP).

Clienti

Nella versione base del problema dell'istradamento dei veicoli si considerano clienti caratterizzati solamente da un quantitativo di merce che deve essere ritirata (o consegnata).
Casi più generali specificano altre caratteristiche dei clienti.
Per esempio, in alcune applicazioni il cliente deve essere servito in un dato intervallo temporale).
Si pensi ad esempio ai negozi che hanno come orario di consegna della merce le ore di apertura del negozio stesso.
Si parla in questi casi di problemi di vehicle routing con time windows (VRPTW).
Una particolarità di questa classe di problemi `e che se il veicolo raggiunge il cliente al di fuori della finestra temporale consentita allora il veicolo deve aspettare (nel caso in cui arrivi prima dell'inizio della finestra temporale) o deve andar via senza portare a termine l'operazione (nel caso in cui sia in ritardo).
Un'altra famiglia di problemi che citiamo sono problemi di pick-up and delivery in cui alcuni clienti richiedono il ritiro di merce, mentre altri richiedono la consegna di merce, e la merce ritirata da un cliente può sia essere consegnata ad un deposito (per essere spedita in un'altra località) oppure essere consegnata ad un altro cliente in zona.
Chiaramente in quest'ultimo caso il ritiro della merce dovrà avvenire prima della sua consegna al cliente finale.
Inoltre, un'altra complicazione di cui si deve tener conto in questa famiglia di problemi `e dovuta alla capacità del veicolo.
Si pensi ad una rotta composta da molti clienti che richiedono il ritiro di merce voluminosa le cui consegne verranno effettuate alla fine della rotta.
Ciò potrebbe risultare in rotte non ammissibili e quindi in fase di soluzione del problema si dovrà tener conto della massima capacità del veicolo e dell'ordine in cui i clienti vengono visitati.

Veicoli

Un'altra dimensione per la classificazione dei problemi di VRP `e data dalle caratteristiche dei veicoli.
Si parla di flotte omogenee quando i veicoli che le compongono sono tutti identici tra di loro, mentre si parla di flotte eterogeneee nel momento in cui i veicoli differiscono tra di loro per diverse caratteristiche (ad esempio capacità di carico, costi operativi, autonomia, velocità ...).
I veicoli non omogenei potrebbero essere caratterizzati (oltre dalla capacità e dall'autonomia) anche da diverse abilità operative; si pensi ad esempio a problemi di raccolta dei rifiuti in centri storici, i veicoli di maggior dimensione non sono in grado di passare nei vicoli di un centro storico medioevale, mentre sono estremamente efficienti in zone più moderne con strade più ampie.
Altra possibile dimensione da considerare per la classificazione dei problemi di VRP è data dalla funzione obiettivo. In alcune versioni si cerca di minimizzare la distanza percorsa dai veicoli, in altre versioni i costi (che possono includere costi fissi e costi varibaili), o il numero di veicoli necessari, giusto per citare alcune varianti.

Rete di trasporto

Si parla di reti euclidee, nel caso in cui le distanze tra i nodi rispettino la diseguaglianza triangolare $c_{ij} \leq c_{ih} + c_{hj}$.
Si parla di istanze planari, quando le distanze tra i clienti sono calcolate a partire dalla disposizione dei clienti in un piano.
Ovvero quando ogni cliente è caratterizzato da due coordinate che ne definiscono la sua posizione nel piano cartesiano e le distanze tra i clienti sono calcolate come le distanze tra i punti del piano.
Inoltre, supporremo che i grafi rappresentanti le infrastrutture di trasporto siano grafi completi. Nel caso in cui un arco di collegamento tra due nodi non dovesse essere presente supporremo la presenza di un costo infinito associato all'arco.
Infine, in alcune applicazioni si considera anche la presenza di più depositi.
Le diverse caratteristiche del nodo deposito danno luogo alle varianti multi-deposito.
Nel caso in cui si abbiano a disposizione più depositi si potrebbero presentare problemi con depositi aventi diverse caratteristiche, come ad esempio la flotta a disposizione o diversi quantitativi di merce da distribuire.

1.2 Classificazione

I problemi di trasporto sono associati ad altri problemi decisionali quali:

- localizzazione dei centri logistici
- allocazione della domanda
- gestione delle scorte

Una molteplicità di problemi decisionali:

- composizione delle flotte
- turni dei veicoli e del personale
- determinazione delle rotte
- assegnazione dei carichi e loro composizione
- posizionamento dei veicoli vuoti

Due classi principali di problemi:

- **problemi di trasporto a lunga distanza**
- **problemi di trasporto a breve distanza**

I problemi di trasporto a lunga distanza (long-haul, intercity freight transportation)
A seconda del contesto in cui si opera i trasporti avvengono per viaggi diretti, o viaggi indiretti:

- Viaggi diretti (singola origine-destinazione)
    - da pochi a molti
    - realizzati in proprio dalle ditte produttrici

- Viaggi indiretti
    - da molti a molti
    - spedizionieri con molteplicità di clienti
    - servizio in linea (orario prestabilito) o su richiesta (allocazione dinamica dei mezzi).

I problemi di trasporto a breve distanza (short-haul, local freight transportation)
Sono caratterizzati da:

• Ambito cittadino o regionale
• Presenza di uno o più depositi
• Arco temporale giornaliero
• Problematiche caratteristiche:

- **organizzazione dell'attività di raccolta e distribuzione** (delivery)
- **definizione delle rotte dei veicoli**
- distribuzione e raccolta combinata (pick-up and delivery)
- domanda variabile (giornaliera)

Formulazione problemi di trasporto a breve distanza

Dato un grafo **G** = (**V** , **A**) generico bisogna determinare l'insieme di **m** cicli (rotte) a costo minimo che comprendano **U** □ **V** nodi di servizio (required vertices) e **R** □ **A** archi di servizio (required edges) dove :
costo di una rotta = somma costi degli archi che la compongono
e che soddisfino ad un insieme dato di vincoli operativi:

- ogni rotta deve passare per uno o più nodi stabiliti
- numero massimo dei veicoli
- rispetto della capacità dei veicoli (peso, volume)
- durata massima delle rotte
- fasce orarie di servizio per i clienti (time windows)
- richieste di servizio dei clienti soddisfatte da un singoli veicolo

E’ necessario valutare la **localizzazione dei clienti,** i quali possono essere:

- distribuiti in modo discontinuo lungo le vie
- concentrati nelle località associate ai nodi: Node Routing Problem (NRP)
- distribuiti in modo continuo lungo le vie: Arc Routing Problem (ARP)

Capacitated Vehicle Routing Problem (CVRP)

Il CVRP - Capacitated Vehicle Routing Problem `e la versione pi`u comune di questa famiglia di problemi.
Ciò che caratterizza questa tipologia di problemi `e il fatto che il servizio sia di semplice consegna senza raccolta.
Inoltre le richieste dei clienti sono note a priori e deterministiche e devono essere soddisfatte da un solo veicolo; tutti i veicoli sono identici e basati su di un singolo deposito centrale.
Gli unici vincoli imposti riguardano le capacità dei veicoli e l’obiettivo è minimizzare il costo totale di servizio, che può essere una funzione del numero dei route, della loro lunghezza complessiva o del tempo di percorrenza.
Il CVRP richiede la determinazione di un insieme di esattamente K circuiti semplici, ognuno corrispondente al percorso di un veicolo, in modo che il costo totale del trasporto, definito dalla somma dei costi espressi sugli archi, sia minimo.
I vincoli del problema sono i seguenti:

- ogni veicolo, e quindi ogni circuito, deve transitare per il deposito;
- ogni cliente `e visitato da uno ed un solo circuito;
- la somma delle richieste di merce dei vertici visitati da ogni circuito non può eccedere la capacità C dei veicoli.

Il CVRP `e un problema che generalizza il ben noto Travelling Salesman Problem (TSP), che richiede di determinare la rotta a costo minimo che tocca tutti i vertici di un grafo G, la cui soluzione per ogni veicolo dunque è una sequenza di nodi.
Un’istanza CVRP si riduce proprio ad un’istanza di TSP

Distance-Constrained Vehicle Routing Problem (DVRP)

E' una importante variante del: travel in questo problema i vincoli di capacità riguardanti ognuno dei route sono sostituiti da vincoli di lunghezza o di tempo massimi.
In particolare una lunghezza non negativa t_{ji} viene associata a ciascun lato o arco **(i,j),** e la lunghezza totale degli archi appartenenti ad un route non può superare un valore massimo definito come **T**.
Quando, invece, i parametri t_{ij} rappresentano tempi di viaggio, ad ogni vertice i può essere assegnato un tempo di servizio s_i, pari al tempo necessario ad un veicolo per compiere il proprio servizio presso il cliente.
In alcuni casi i tempi di servizio possono essere inclusi nei costi temporali dei lati, ponendo per ogni arco **(i, j)** $t_{ij} = t_{0ij} + s_i/2 + s_j/2$, dove t_{0ij} è il costo temporale per la sola percorrenza dell'arco **(i, j).**

Distance-Constrained Capacitated VRP (DCVRP)

In questa seconda variante sono imposte entrambe le famiglie di vincoli:
ogni route ha una lunghezza o un tempo di percorrenza massimo e, nel contempo, il veicolo che lo percorre ha una limitata capacità di trasporto.
In genere le matrici dei costi e delle distanze coincidono, vale cioè l'uguaglianza
$c_{ij} = t_{ij}$ per tutti gli archi **(i, j) € A**. L'obiettivo del problema corrisponde allora a minimizzare la lunghezza totale dei route oppure, se il tempo di servizio è incluso nei costi temporali degli archi, la loro durata.

VRP con Pickup and Delivery (VRPPD)

Nella versione di base del VRP con Pickup and Delivery (VRPPD), ogni Cliente **i** è associato a due quantità non negative d_i e p_i, rappresentanti la richiesta di merce e la quantità della stessa da ritirare rispettivamente.
Talvolta può essere memorizzato per comodità solamente un parametro, $d_i - p_i$, che rappresenta la differenza netta di merce necessaria (eventualmente negativa).
Per ogni vertice **i**, inoltre, sono presenti altri due parametri, O_i e D_i, che caratterizzano così il problema: la merce richiesta dal vertice i deve essere preventivamente raccolta

dal veicolo presso il cliente $\mathbf{O_i}$. Allo stesso modo, la merce ritirata presso il cliente i deve essere consegnata al cliente $\mathbf{D_i}$ che deve quindi essere visitato successivamente.
Per convenzione si assume che lo scarico della merce avvenga sempre prima del caricamento.
Un problema di VRPPD consiste perciò nel determinare **K** route di costo minimo e tali che:

- ogni circuito visiti il deposito;
- ogni cliente sia visitato da uno e un solo circuito;
- il carico dei veicoli, in ogni punto del route, sia non negativo e non ecceda la capacità totale C;
- per ogni cliente **i**, il vertice $\mathbf{O_i}$, se diverso dal deposito, venga visitato nello stesso circuito e prima della visita di **i**;
- per ogni cliente **i**, il vertice $\mathbf{D_i}$, se diverso dal deposito, venga visitato nello stesso circuito e dopo della visita di **i**;

Spesso l'origine $\mathbf{O_i}$ e la destinazione $\mathbf{D_i}$ coincidono per tutti i vertici -possono corrispondere, ad esempio, con il deposito - e non sono indicati esplicitamente.

VRP con Time Window (VRPTW)

Il VRP con Time Window (VRPTW) `e un'altra estensione del CVRP in cui ad ogni cliente **i** è associato un intervallo di tempo $\mathbf{[\ a_i, b_i\ ]}$ detto, appunto, time window.
Il servizio di ogni cliente deve iniziare in un istante $\mathbf{t_i}$ contenuto nel time window; in caso di arrivo anticipato al vertice **i** il veicolo rimane fermo attendendo il tempo ai perché possa quindi essere effettuato il servizio.
Ogni cliente `e associato ad un tempo di servizio $\mathbf{s_i}$, che rappresenta la durata dell'intervallo di tempo durante il quale il veicolo che effettua il servizio staziona presso il cliente.
Altri dati del problema sono la matrice dei tempi di viaggio, la cui generica entry $\mathbf{t_{ij}}$ è pari al tempo di percorrenza dell'arco **(i, j) € A**, e $\mathbf{t_0}$, l'istante di tempo nel quale i veicoli lasciano il deposito.
Usualmente le matrici di costi e tempi coincidono e si suppone che tutti i veicoli partano dal deposito all'istante $\mathbf{t_0 = 0}$.

VRPTW `e di norma rappresentato come un problema asimmetrico, in quanto i valori dei time window inducono implicitamente un orientamento dei route.
Riassumendo la soluzione di un'istanza di VRPTW consiste nella determinazione di K circuiti semplici di costo minimo tali che :

- ogni circuito visiti un deposito;
- ogni cliente sia visitato da esattamente un circuito;
- la somma delle richieste dei clienti visitati da un route non ecceda la capacità C del veicolo che li serve;
- per ogni cliente **i**, il servizio abbia inizio in un istante compreso nel time window **[a_i, b_i]** e il veicolo rimanga occupato per un tempo pari a **s_i**.

Si può definire un modello matematico per il VRPTW assumendo validi tutti i vincoli già esplicitati se e introducendo una nuova variabile decisionale **y_{ki}** rappresentante il tempo in cui il veicolo **k € K** inizia il servizio presso il cliente **i**-esimo.
I nuovi vincoli aggiuntivi sono perciò:

$$x_{kij} \; (\; y_{ki} + t_{ij} - y_{kj} \;) \leq 0$$

per ogni **(i, j) € A** , per ogni **k € K**

Tale vincolo impone che il veicolo **k** non possa arrivare a **j** prima di $y_{ki} + t_{ji}$ se percorre l'arco da **i** a **j.**

$$a_i \leq y_{ki} \leq b_i$$

per ogni **(i) ,** per ogni **k € K**

Tale vincolo assicura che ogni time window sia rispettata.

CAPITOLO 2

Pianificazione ed instradamento del veicolo con vincoli di capacità e finestre temporali (VRPTW): applicazione specifica per la raccolta ed il trasporto di pelli bovine dai macelli alla conceria.

2.1 Ambiente e dichiarazione del problema.

Si consideri un territorio dove esiste una rete stradale ove sono dislocati in posizioni fisse:

- Una conceria
- Dei macelli
- Uno o più depositi per i camion

I macelli forniscono pelli bovine fresche che devono essere trasportate alla conceria tramite di camion dedicati, rispettando i vincoli di capacità e delle finestre temporali.
I mezzi, ognuno caratterizzato da una capacità massima di peso, sono situati nei depositi; ogni mezzo utilizzato inizia dal suo stesso deposito rispettando la sua finestra temporale di partenza e implicando ad ogni partenza un costo fisso.
Ogni macello fornisce una quantità specifica di pelle bovina fresca; queste pelli vengono caricate su un mezzo rispettando una data finestra temporale, implicando un dato tempo di carico e costo di carico; per alcuni macelli il peso delle pelli fornite può essere nullo.
Un mezzo può muoversi dal suo stesso deposito per raggiungere il primo macello, caricare tutte le pelli fornite dal macello rispettando la propria finestra temporale e impiegando un certo tempo e costo di carico, muoversi verso il secondo macello e caricare, verso il terzo e caricare, e così via fino all'ultimo macello presente sul suo percorso, per poi andare fino alla conceria e scaricare tutte le pelli caricate.
L'operazione di scarico richiede un tempo e un costo fisso e deve rispettare la finestra temporale della conceria, inoltre per ogni macello, se il camion ha caricato la pelle, è necessario rispettare un tempo massimo di scarico in conceria per preservare la freschezza della pelle.

La pelle fresca infatti richiede di essere operata a di essere portata a bassa temperatura in tempi brevi (ore) dopo l'uccisione dell'animale, altrimenti rimarrebbe irreparabilmente danneggiata.costo fisso
Pause durante il percorso del mezzo sono permesse per rispettare le finestre temporali.
Il peso totale delle pelli caricate sul mezzo non può superare la sua capacità massima.
Vengono forniti il tempo e i costi richiesti per lo spostamento dal deposito per ogni macello e per ogni mezzo.
Per ogni macello vengono forniti costo e tempo per lo spostamento da questo al successivo e per raggiungere la conceria.
E' richiesto un costo aggiuntivo, proporzionale al tempo totale del percorso; inoltre è considerato un costo associato ad un eccessivo ritardo alla partenza.
Il problema consiste nel definire il percorso che ogni mezzo dovrà eseguire (c'è la possibilità che alcuni mezzi rimangano fermi al deposito) in modo da raccogliere le pelli fornite da tutti i macelli (tralasciando i macelli con fornitura nulla), rispettando le capacità e le finestre temporali dei vincoli, con il minimo costo totale.

2.2 Modello di programmazione lineare

Si abbiano:

- **r** mezzi indicate con **k**
- **n** macelli indicate da **i** o **j**
- una conceria chiamata **l**.

I depositi sono associati ai mezzi, come indicato sotto.

Ogni mezzo **k**, **k=1**, **r** è caratterizzato da:

- una capacità $\mathbf{q_k}$
- un costo di uscita fisso $\mathbf{c_k}$, che deve essere pagato se il mezzo compie qualsiasi percorso (ad esempio se non rimane al suo stesso deposito)

- un costo unitario di tempo $\mathbf{ct_k}$ pagato per ogni unità di tempo spesa tra l'uscita dal deposito e la fine dello scarico in conceria
- un costo unitario di tempo $\mathbf{cd_k}$ pagato per ogni unità di tempo spesa ritardando l'uscita dal deposito
- un deposito $\mathbf{m_k}$ e la relativa finestra temporale $\mathbf{[s_{mk}, f_{mk}]}$
- il tempo richiesto per scaricare le pelli alla conceria $\mathbf{t_{lk}}$.
- Il costo richiesto per scaricare le pelli alla conceria $\mathbf{d_{tlk}}$.

Ogni macello **i**, **i = 1**, **n** è caratterizzato da:

- il peso totale delle pelli da fornire $\mathbf{p_i}$.
- la finestra temporale di carico $\mathbf{[s_i, f_i]}$.
- un tempo massimo $\mathbf{g_i}$ prima del quale tutte le pelli fornite devono essere scaricate in conceria per la conservazione.

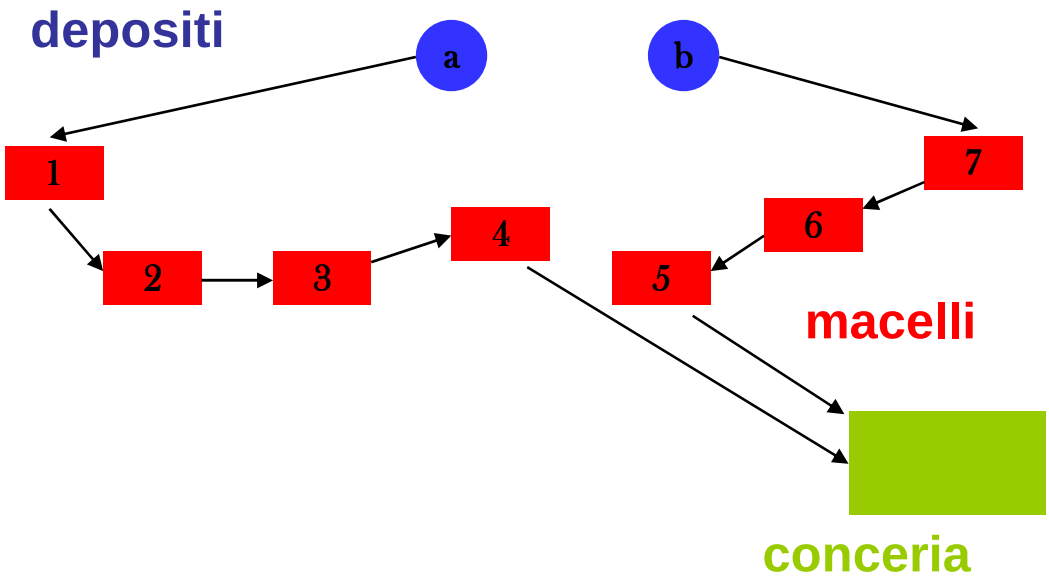
Ogni coppia macello-mezzo **ik, i=1, n, k=1, r** è caratterizzato da:

- il tempo di carico $\mathbf{t_{ik}}$ che permette al mezzo **k** di caricare le pelli fornite dal macello **i**
- il tempo speso per spostare il mezzo **k** dal deposito al macello **i**, $\mathbf{t_{mik}}$.
- il tempo speso per spostare il mezzo **k** dal macello **i** alla conceria $\mathbf{t_{ilk}}$.
- il costo richiesto per spostare il mezzo **k** dal deposito al macello **i**, $\mathbf{d_{mik}}$.
- il costo richiesto per spostare il mezzo **k** dal macello **i** alla conceria $\mathbf{d_{ilk}}$.
- Il costo di carico per caricare il mezzo **k** presso il macello **i**, $\mathbf{d_{tik}}$.

Ogni tripla macello-macello-mezzo **ijk, i,j=1, n, k=1, r** è caratterizzato da:

- il tempo speso per spostare il mezzo **k** dal macello **i** al macello **j**, $\mathbf{t_{ijk}}$.
- il costo richiesto per spostare il mezzo **k** dal macello **i** al macello **j**, $\mathbf{d_{ijk}}$.

In fine un grande numero **M** è utilizzato nelle equazioni quando necessario.



I vincoli possono essere classificati come “ vincoli di percorso ” e “ vincoli di tempo ”.

Vincoli di percorso :

- $\sum(x_{mik} \mid i=1,n) + x_{mlk} = 1$ $k=1,r$
  Ogni camion può partire dal suo stesso deposito o rimanere li.
- $x_{mik} + \sum(x_{jik} \mid j=1,n, j\neq i) = \sum(x_{ijk} \mid j=1,n, j\neq i) + x_{ilk}$ $i=1,n, k=1,r$
  ogni macello con disponibilità di pelle fresca deve essere servitor da un solo camion.
- $\sum(q_k[x_{mik}+\sum(x_{jik} \mid j=1,n, j\neq i)] \mid k=1,r) \geq p_i$ $i=1,n$
  Il peso delle pelli fresche disponibili presso un macello deve essere strettamente positivo.
- $\sum(p_i\, x_{jik} \mid i,j=1,n, j\neq i) + \sum(p_i\, x_{mik} \mid i=1,n) \leq q_k$ $k=1,r$
  Il carico totale di ogni camion deve essere inferiore alla sua capacità massima.

Vincoli di tempo:

- $z_{mk} \geq s_{mk}$ $k=1,r$
  ogni camion non può partire dal suo stesso deposito prima dell’inizio della relative finestra temporale.

- $z_{mk} \leq s_{fk} - [s_{fk} - s_{mk}]\, x_{mlk}$ $\quad k=1,r$

  ogni camion che lascia il proprio deposito non può partire dopo la fine della relativa finestra temporale.

- $z_{ik} \geq z_{mk} + t_{mik} + t_{ik} - M\,[1 - x_{mik}]$ $\quad i=1,n,k=1,r$

  questo è il minimo tempo richiesto perché un camion possa spostarsi dal proprio deposito ad un macello.

- $z_{ik} \geq z_{jk} + t_{jik} + t_{ik} - M\,[1 - x_{jik}]$ $\quad i,j=1,n,j\neq i,\ k=1,r$

  questo è il minimo tempo richiesto perché un camion sia in grado di spostarsi da un macello al successivo.

- $z_{ik} \geq s_i + t_{ik} - M\,[1 - x_{mik} - \sum(x_{jik} \mid j=1,n,\ j\neq i)]$ $\quad i=1,n,\ k=1,r$

  il tempo richiesto affinchè un camion possa spostarsi da un macello deve rispettare l'inizio della finestra temporale del macello.

- $z_{ik} \leq f_i + M\,[1 - x_{mik} - \sum(x_{jik} \mid j=1,n,\ j\neq i)]$ $\quad i=1,n,k=1,r$

  il tempo richiesto affinchè un camion possa spostarsi da un macello deve rispettare la fine della finestra temporale del macello.

- $z_{lk} \geq z_{ik} + t_{ilk} + t_{lk} - M\,[1 - x_{ilk}]$ $\quad i=1,n,k=1,r$

  questo è il minimo tempo necessario ad un camion per scaricare presso la conceria.

- $z_{lk} \geq s_l + t_{lk}$ $\quad k=1,r$

  il tempo alla fine dell'operazione di scarico deve rispettare l'inizio della finestra temporale della conceria.

- $z_{lk} \leq f_l$ $\quad k=1,r$

  il tempo alla fine dell'operazione di scarico deve rispettare la fine della finestra temporale della conceria.

- $z_{lk} \leq g_i + M\,[1 - x_{mik} - \sum(x_{jik} \mid j=1,n,\ j\neq i)]$ $\quad i=1,n,k=1,r$

  il tempo alla fine dell'operazione di scarico deve rispettare il tempo massimo imposto dai macelli, entro il quale le pelli devono essere scaricate per la conservazione.

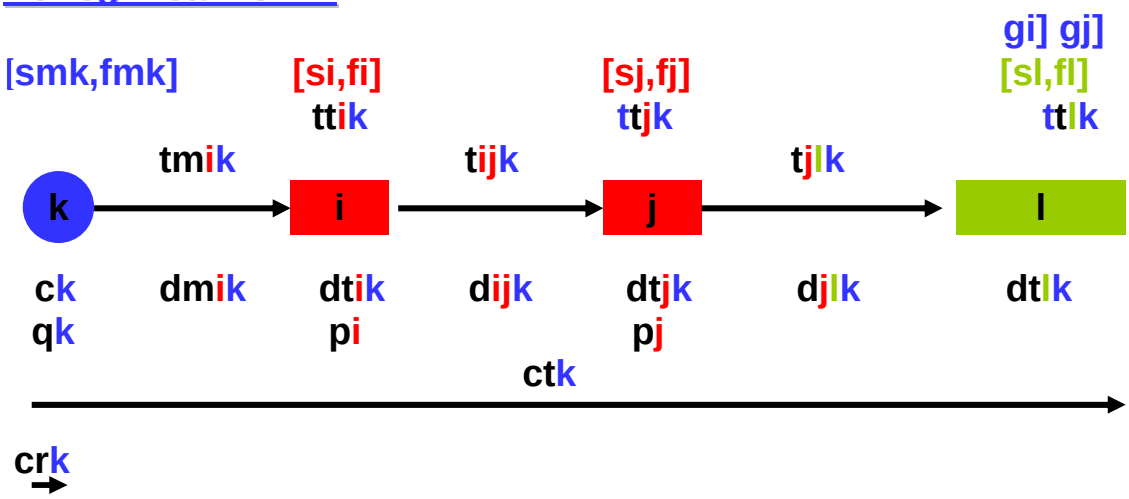
I tempi e costi di carico sono forniti per ogni coppia macello-mezzo.

I tempi e costi di scarico presso la conceria sono forniti per ogni camion.

I costi e i tempi di trasporto e i costi di viaggio sono forniti per ogni camion in ogni connessione

- Deposito-macello
- Macello-macello
- Macello-conceria

Per ogni camion k:



Il problema oggettivo è dato dalla minimizzazione dei costi totali, ovvero della sommatoria in **k** :

MINIMIZZARE:

$\sum_k$ { **Costo d'uscita + Costo totale di trasporto + Costo totale di carico + Costo di scarico + Costo totale tempo di viaggio + Costo ritardo alla partenza** }

Cioè:

$$\text{MIN}\{\sum (c_k\, x_{mik} \mid i=1,n,\ k=1,r) + \sum(d_{mik}\, x_{mik} \mid i=1,n,\ k=1,r) + \sum(d_{ijk}\, x_{ijk} \mid i,j=1,n,\ i\neq j,\ k=1,r) + \sum(d_{ilk}\, x_{ilk} \mid i=1,n,\ k=1,r) + \sum(ct_k\,[z_{lk} - z_{mk}] \mid k=1,r) + \sum(cd_k\,[z_{mk} - s_{mk}] \mid k=1,r)\}$$

s.t.:

$$\sum(x_{mik} \mid i=1,n) + x_{mlk} = 1 \qquad k=1,r$$

$$x_{mik} + \sum(x_{jik} \mid j=1,n,\ j\neq i) = \sum(x_{ijk} \mid j=1,n,\ j\neq i) + x_{ilk} \qquad i=1,n,\ k=1,r$$

$$\sum(q_k[x_{mik}+\sum(x_{jik} \mid j=1,n,\ j\neq i)] \mid k=1,r) \geq p_i \qquad i=1,n$$

$$\sum(p_i\, x_{jik} \mid i,j=1,n,\ j\neq i) + \sum(p_i\, x_{mik} \mid i=1,n) \leq q_k \qquad k=1,r$$

$$z_{mk} \geq s_{mk} \qquad k=1,r$$

$$z_{mk} \leq s_{fk} - [s_{fk} - s_{mk}]\, x_{mlk} \qquad k=1,r$$

$$z_{ik} \geq z_{mk} + t_{mik} + t_{ik} - M\,[1 - x_{mik}] \qquad i=1,n,\ k=1,r$$

$$z_{ik} \geq z_{jk} + t_{jik} + t_{ik} - M\,[1 - x_{jik}] \qquad i,j=1,n,\ j\neq i,\ k=1,r$$

$$z_{ik} \geq s_i + t_{ik} - M\,[1 - x_{mik} - \sum(x_{jik} \mid j=1,n,\ j\neq i)] \qquad i=1,n,\ k=1,r$$

$$z_{ik} \leq f_i + M\,[1 - x_{mik} - \sum(x_{jik} \mid j=1,n,\ j\neq i)] \qquad i=1,n,\ k=1,r$$

$$z_{lk} \geq z_{ik} + t_{ilk} + t_{lk} - M\,[1 - x_{ilk}] \qquad i=1,n,\ k=1,r$$

$$z_{lk} \geq s_l + t_{lk} \qquad k=1,r$$

$$z_{lk} \leq f_l \qquad k=1,r$$

$$z_{lk} \leq g_i + M\,[1 - x_{mik} - \sum(x_{jik} \mid j=1,n,\ j\neq i)] \qquad i=1,n,\ k=1,r$$

$$x_{mlk} \in \{0,1\} \qquad k,\ k=1,r$$

$$x_{mik} \in \{0,1\} \qquad i=1,n,\ k=1,r$$

$$x_{ilk} \in \{0,1\} \qquad i=1,n,\ k=1,r$$

$$x_{ijk} \in \{0,1\} \qquad i,j=1,n,\ k=1,r$$

$$z_{mk} \geq 0 \qquad k=1,r$$

$$z_{ik} \geq 0 \qquad i=1,n,\ k=1,r$$

$$z_{lk} \geq 0 \qquad k=1,r$$

2.3 Visual Basic.NET o Visual basic

Tra i vari metodi risolutivi “esatti “ per la risoluzione del VRP, ho utilizzato Visual basic (formalmente abbreviato **VB**), un linguaggio di programmazione event driven arrivato fino alla versione 10, la cui sintassi deriva dal BASIC.

Con l'uscita del .NET Framework (progetto all'interno del qualeMicrosoft ha creato una piattaforma di sviluppo software,.NET, la quale è una versatile tecnologia di programmazione ad oggetti.) Microsoft ha introdotto una nuova "visione" del suo più famoso linguaggio di sviluppo. Non è, perciò, un aggiornamento del vecchio VB6, ma un nuovo linguaggio che integra tutte le classi di base del framework .NET, mantenendo una sintassi simile a quella di Visual Basic.

Sono state introdotte tutta una serie di funzionalità tipiche del .NET Framework, come "remoting", "Web Service", "Servizi Windows", ADO.NET e il completo supporto alla programmazione a oggetti (ereditarietà, polimorfismo, ecc.).

Anche se esiste un tool per la migrazione delle applicazioni, alcune funzionalità non possono essere convertite automaticamente, richiedendo quindi uno sforzo successivo per completare la migrazione.

Molte delle originali carenze sono state colmate, ma questo ha reso Visual Basic .Net un linguaggio più complesso del suo predecessore, al tempo stesso estremamente più potente e versatile.

Inoltre, essendo un'archiettura basata sul .NET Framework, i linguaggi condividono le stesse identiche potenzialità; di conseguenza, la scelta di utilizzare l'uno o l'altro linguaggio è una cosa influenzata più dalla cultura degli sviluppatori che dalle reali necessità tecniche.

Analizzando la presenza di Visual Basic .NET in Internet è chiaro che questo linguaggio rimane tuttora il punto di riferimento e la prima scelta della maggior parte dei programmatori.

Caratteristiche

- Facilità d'uso (non utilizza formalità di punteggiatura tipica di quasi tutti gli altri linguaggi);
- il suo ambiente di lavoro RAD (metodologia di sviluppo del software che coinvolge modelli di sviluppo iterativi, la costruzione di prototipi e l'utilizzo di

strumenti CASE e che comporta compromessi tra usabilità, funzionalità e velocità d'esecuzione) che permette di realizzare in breve tempo interfacce GUI anche complesse (tipo di interfaccia utente che consente all'utente di interagire con la macchina manipolando oggetti grafici convenzionali).

- il pratico accesso alle basi dati;
- la creazione di controlli activeX (tecnologia sviluppata dalla Microsoft destinata agli sviluppatori e creata per poter estendere le potenzialità e le funzioni di un'applicazione permettendo di aggiungere nuove possibilità, comandi ed, eventualmente, semplificando alcuni processi nell'ambito dello sviluppo di software) con il linguaggio stesso.
- Tramite l'integrazione dei controlli VBX (nelle versioni a 16 bit), dei controlli OCX (presenti nelle versioni a 32 bit) e dei collegamenti OLE presenti nell'ambiente di lavoro o realizzati da altri programmatori, è possibile aggiungere potenzialità al linguaggio, aggiungendo nei propri progetti nuove funzioni o ampliando funzioni già esistenti.
-

Carenze e vantaggi

Molti programmatori hanno un rapporto difficile con Visual Basic in quanto è stato progettato per essere un linguaggio semplice e molte caratteristiche che sono presenti in linguaggi come Java non sono disponibili in Visual Basic.

Nell'interesse della convenienza e del rapido sviluppo, alcune funzionalità come la verifica del "cast" dei dati e la dichiarazione obbligatoria delle variabili (anche se è possibile rendere questa dichiarazione obbligatoria) sono disabilitate.

Questo rende Visual Basic molto semplice da utilizzare, ma i programmi così realizzati possono essere distribuiti con errori nascosti dovuti a questa mancanza di controlli.

Molti critici di Visual Basic spiegano che la sua natura semplice è dannosa nel lungo periodo.

Innanzitutto, è spesso utilizzato per insegnare la programmazione proprio a causa della sua semplicità, ma imparare a programmare in Visual Basic non introduce i programmatori a molte delle tecniche e dei costrutti fondamentali della programmazione.

In secondo luogo, come già detto in precedenza, la possibilità di disabilitare errori e avvertimenti in fase di compilazione, può rendere difficoltosa la ricerca degli errori.

Molti dei suoi sostenitori insistono, invece, sul fatto che la sua semplice natura è il suo punto di forza, in quanto permette un rapido sviluppo di applicazioni da parte di programmatori esperti e una curva di apprendimento piuttosto breve per i programmatori che provengono da altri linguaggi.
Inoltre, le applicazioni Visual Basic possono essere integrate con le basi dati in maniera molto semplice.
Visual Basic è, inoltre, un agglomerato di funzionalità e sintassi diverse con meno consistenza, ma con più tolleranza degli altri moderni linguaggi di programmazione. Alcune istruzioni come "gosub", "on error" e la dichiarazione del tipo di variabile con l'ultimo carattere del nome (ad esempio miaStringa$ per le stringhe), sono retaggi delle origini del Basic che sono state incluse per avere retrocompatibilità.
Il linguaggio continua ad attirare molti apprezzamenti e molte critiche, ma la grande base "storica" di programmatori che lo hanno utilizzato e che basano su di esso la propria produttività fa sì che continui ad avere una grande diffusione, nonostante l'uscita del successore Visual Basic .Net.
E' bene ricordare che Visual Basic .NET non è una semplice evoluzione di VB6, ma è a tutti gli effetti un altro linguaggio, ad oggetti, basato su classi, costruttori e forti tipizzazioni che lo rendono un sofisticato strumento di sviluppo per la nuova generazione di software e di sistemi operativi.

Tra le varie versioni di Visual Basic. Net, ho utilizzato **Visual Basic 2010 (VB 10)**, l'ultima versione disponibile del linguaggio che è stata rilasciata nel mese di aprile 2010 insieme al Framework .NET 4.0 e al Visual Studio 2010.

Una delle nuove funzionalità più rilevanti è il supporto a Dynamic Language Runtime (DLR), presente nel Framework 4.0 attraverso la classe System.Dynamic[1]. Inoltre, introduce ulteriori caratteristiche sintattiche che lo rendono più *pulito*, come la possibilità di spezzare le linee di codice su più righe senza dover specificare il carattere underscore (anche se con alcune limitazioni) e aggiunge importanti modifiche ai generics (varianza e covarianza) e le Parallel Extensions per lo sviluppo multi-threading.

CAPITOLO 3

Implementazione del programma risolutivo del VRPTW in Visual Basic .

3.1 Classi

Le classi che ho trovato necessario creare sono:

- la classe **connect**
- la classe **mezzo**
- la classe **depositi**
- la classe **macelli**
- la classe **gestione dati**

con i rispettivi attributi e metodi che saranno condivisi da tutti gli oggetti creati (*istanze*).

La classe connect

è costituita da tre metodi:

- GeneraStringaConnessioneAccess

```
Private Const Connessione As String =
"provider=Microsoft.Jet.OLEDB.4.0;Data Source=dbtest.mdb;Persist
Security Info=False"
    Private Function GeneraStringaConnessioneAccess() As String
        Dim OutValue As String = Connessione
        Return OutValue
    End Function
```

- EseguiQueryDati

```
Public Sub EseguiQueryDati(ByVal query As String)
        Dim connection As New
OleDbConnection(GeneraStringaConnessioneAccess)
```

```
        Try
            Dim cmd As New OleDbCommand(query, connection)
            connection.Open()
            ' Eseguo l'istruzione sql
            cmd.ExecuteNonQuery()
        Catch ex As Exception
            Dim e As String = ex.Message
        Finally
            'Chiudo la connessione con il Database Rubrica
            connection.Close()
        End Try
    End Sub
```

- Leggidati

```
Public Function Leggidati(ByVal query As String) As DataSet
        Dim DS As System.Data.DataSet
        Dim MyCommand As System.Data.OleDb.OleDbDataAdapter
        Dim connection As New
OleDbConnection(GeneraStringaConnessioneAccess)
        MyCommand = New
System.Data.OleDb.OleDbDataAdapter(query, connection)
        DS = New System.Data.DataSet()
        MyCommand.Fill(DS)
        connection.Close()
        Return DS
    End Function
```

Questi tre metodi nel loro insieme consentono :

- la connessione ad uno o più database in Access
- l'interrogazione del database (operazioni di selezione, inserimento, cancellazione dati)
- la lettura del database

La classe mezzo

Vengono definiti i seguenti attributi:

```
Public nome As String
    Public qk As String
    Public ck As String
```

```
    Public ctk As String
    Public cdk As String
    Public dep As String
    Public tk As String
```

- nome ➔ codice del mezzo
- qk ➔ carico massimo del mezzo
- ck ➔ costo fisso d'uscita
- ctk ➔ costo per unità di tempo
- cdk ➔ costo per unità di tempo di ritardo alla partenza
- dep ➔ codice deposito corrispondente
- tk ➔ tempo di scarico in conceria

e i seguenti metodi:

- NuovoMezzo

```
Public Function NuovoMezzo(ByVal nome As String)
        Dim cn As New connessione.connect
        Dim str As String = ""

        'INSERT
        str = "insert into mezzo "
        str &=
"(Codice,capacita,costouscita,costotrasporto,costoritardo,tscari
co,deposito)"
        str &= "values"
        str &= "('" & nome & "','" & qk & "','" & ck & "','" &
ctk & "','" & cdk & "','" & tk & "','" & dep & "')"
        cn.EseguiQueryDati(str)
        Return True
    End Function
```

Inserisce nella tabella "mezzo" i valori degli attributi dell'oggetto considerato (chiave ➔ codice)

- Salva Mezzo

```
Public Function SalvaMezzo(ByVal codice As String) As Boolean
```

```
        Dim cn As New connessione.connect
        Dim str As String = ""
        'UPDATE
        str = "Update mezzo set "
        str &= "Codice='" & nome & "',"
        str &= "capacita='" & qk & "',"
        str &= "costouscita='" & ck & "',"
        str &= "costotrasporto='" & ctk & "',"
        str &= "costoritardo='" & cdk & "',"
        str &= "tscarico='" & tk & "',"
        str &= "deposito='" & dep & "' "
        str &= " WHERE "
        str &= "Codice='" & codice & "'"
        cn.EseguiQueryDati(str)
        Return True
    End Function
```

Aggiorna i valori degli attributi di un oggetto già esistente nel database.

- CaricaMezzo

```
Public Shared Function CaricaMezzo(ByVal codice As String) As
Mezzi.Mezzo
        Dim cn As New connessione.connect
        Dim ds As New DataSet
        Dim str As String = ""
        Dim m As Mezzi.Mezzo
  seleziona dalla tabella mezzo tutti i codici uguale a 'codice'
        str = "select * from mezzo where codice='" & codice &
"'"     ds = cn.Leggidati(str)
        ' se il numero di righe prelevato dal db è > 0
        If ds.Tables(0).Rows.Count > 0 Then
            'metto nella variabile m l'intera riga
            m = importRow(ds.Tables(0).Rows(0))
        End If
        'chiusura connessione db
        ds.Dispose()
        Return m
```

Permette di visualizzare i valori dei campi del mezzo selezionato.

- CaricaMezzi

```
Public Shared Function CaricaMezzi(Optional ByVal codice As
String = "") As List(Of Mezzi.Mezzo)
        Dim cn As New connessione.connect
        Dim ds As New DataSet
        Dim str As String = ""
        Dim str2 As String = ""
        Dim i As Integer

        Dim Lm As New List(Of Mezzi.Mezzo)  'crea una variabile
array
        'selezioniamo tutto dalla tabella mezzo
        If codice <> "" Then
            str2 = " WHERE codice='" & codice & "'"
        End If
        str = "select * from mezzo " & str2
        ds = cn.Leggidati(str)
        'controllo se il numero di righe prelevate dal db è >0
        If ds.Tables(0).Rows.Count > 0 Then
        'per ogni elemento prelevato dalla query lo inserisco
nell'array Lm
            For i = 0 To ds.Tables(0).Rows.Count - 1
                Lm.Add(importRow(ds.Tables(0).Rows(i)))
            Next
        End If
        ds.Dispose()
        Return Lm
    End Function
```

Carica all'interno di una lista tutti i mezzi presenti nel database che presentano un determinato codice.

- CaricaMezziDt

```
Public Shared Function CaricaMezziDt() As DataTable
        Dim cn As New connessione.connect
        Dim ds As DataSet
        Dim dt As New DataTable
        Dim str As String = ""
        Dim i As Integer
        Dim Lm As New List(Of Mezzi.Mezzo)'   crea una variabile
array
        'selezioniamo tutto dalla tabella mezzo
        str = "select * from mezzo "
        ds = cn.Leggidati(str)
        'controllo se il numero di righe prelevate dal db è >0
```

```
        If ds.Tables(0).Rows.Count > 0 Then
            dt = ds.Tables(0)
        End If
        ds.Dispose()
        Return dt
    End Function
```

Restituisce per mezzo di un datatable (memoria temporanea per la lettura di dati da un database) tutte le proprietà di tutti i mezzi presenti nella tabella “mezzo”.

- importRow

```
    Shared Function importRow(ByVal dr As DataRow) As Mezzi.Mezzo
        Dim m As New Mezzi.Mezzo

        m.nome = dr.Item("codice").ToString
        m.qk = dr.Item("capacita").ToString
        m.ck = dr.Item("costouscita").ToString
        m.ctk = dr.Item("costotrasporto").ToString
        m.cdk = dr.Item("costoritardo").ToString
        m.tk = dr.Item("tscarico").ToString
        m.dep = dr.Item("deposito").ToString

        Return m
    End Function
```

Associa i valori prelevati dall'i-esima riga della tabella “mezzo” alla variabile m di tipo mezzo.

- EsisteCodice

```
    Public Shared Function EsisteCodice(ByVal codice As String)
As Boolean

        Dim cn As New connessione.connect
        Dim ds As New DataSet
        Dim str As String = ""
        Dim ce As Boolean = False
        str = "select codice from mezzo where codice='" & codice
& "'"
        ds = cn.Leggidati(str)
        If ds.Tables(0).Rows.Count > 0 Then
            ce = True
```

```
        End If
        ds.Dispose()
        Return ce
    End Function
```

Verifica l'esistenza di un codice specifico all'interno della tabella mezzo.

- DeleteMezzo

```
Shared Function DeleteMezzo(ByVal codiceold As String) As
Boolean
        Dim cn As New connessione.connect
        Dim str As String = ""
        'Delete
        str = "Delete  from mezzo "
        str &= " WHERE "
        str &= "Codice='" & codiceold & "'"
        cn.EseguiQueryDati(str)
        Return True
    End Function
```

Permette l'eliminazione dal database del mezzo d'interesse.

La classe deposito

Vengono definiti i seguenti attributi:

```
Public codice As String
Public smk As String
Public fmk As String
```

- Codice ➔ codice del deposito
- Smk ➔ l'inizio della finestra temporale del deposito
- Fmk ➔ la fine della finestra temporale del deposito

E i seguenti metodi

- NuovoDeposito

```
Public Function NuovoDeposito(ByVal codice As String)
        Dim cn As New connessione.connect
        Dim str As String = ""

        'INSERT
        str = "insert into deposito "
        str &= "(codice,Tstart,Tfinish)"
        str &= "values"
        str &= "('" & codice & "','" & smk & "','" & fmk & "')"
        cn.EseguiQueryDati(str)
        Return True
    End Function
```

Inserisce nella tabella “deposito” i valori degli attributi dell’oggetto considerato (chiave ➔ codice)

- Salva Deposito

```
Public Function SalvaDeposito(ByVal codice As String) As Boolean
        Dim cn As New connessione.connect
        Dim str As String = ""
        'UPDATE
        str = "Update deposito set "
        str &= "codice='" & codice & "',"
        str &= "Tstart='" & smk & "',"
        str &= "Tfinish='" & fmk & "' "
        str &= " WHERE "
        str &= "codice='" & codice & "'"
        cn.EseguiQueryDati(str)
        Return True
    End Functiona
```

Aggiorna i valori degli attributi di un oggetto già esistente nel database.

- caricaDeposito

```
Public Shared Function CaricaDeposito(ByVal codice As String) As
Depositi.Deposito
        Dim cn As New connessione.connect
        Dim ds As New DataSet
        Dim str As String = ""
        Dim m As Depositi.Deposito
 'seleziona dalla tabella mezzo tutti i codici uguale a 'codice'
```

```
        str = "select * from deposito where codice='" & codice &
"'"     ds = cn.Leggidati(str)
        ' se il numero di righe prelevato dal db è > 0
        If ds.Tables(0).Rows.Count > 0 Then
            'metto nella variabile m l'intera riga
            m = importRow(ds.Tables(0).Rows(0))
        End If 'chiusura connessione db
        ds.Dispose()
        Return m
    End Function
```

Restituisce i valori letti dalla tabella “deposito” che hanno un determinato codice.

- Carica Depositi

```
Public Shared Function CaricaDepositi() As List(Of
Depositi.Deposito)
        Dim cn As New connessione.connect
        Dim ds As New DataSet
        Dim str As String = ""
        Dim i As Integer

        Dim Lm As New List(Of Depositi.Deposito)      'crea una
variabile array
        'selezioniamo tutto dalla tabella deposito
        str = "select * from deposito "
        ds = cn.Leggidati(str)
controllo se il numero di righe prelevate dal db è > 0
        If ds.Tables(0).Rows.Count > 0 Then
            'per ogni elemento prelevato dalla query lo
inserisco nell'array Lm
            For i = 0 To ds.Tables(0).Rows.Count - 1
                Lm.Add(importRow(ds.Tables(0).Rows(i)))
            Next
        End If
        ds.Dispose()
        Return Lm
    End Function
```

Carica all’interno di una lista tutti i depositi presenti nel database che presentano un determinato codice.

- CaricaDepositiDt

```
Public Shared Function CaricaDepositiDt() As DataTable
```

```
        Dim cn As New connessione.connect
        Dim ds As DataSet
        Dim dt As New DataTable
        Dim str As String = ""
        Dim i As Integer
'crea una variabile array
        Dim Lm As New List(Of Depositi.Deposito)
        'selezioniamo tutto dalla tabella deposito
        str = "select * from deposito "
        ds = cn.Leggidati(str)
controllo se il numero di righe prelevate dal db è >0
        If ds.Tables(0).Rows.Count > 0 Then
            dt = ds.Tables(0)
        End If
        ds.Dispose()
        Return dt
        End Function
```

Restituisce per mezzo di un datatable (memoria temporanea per la lettura di dati da un database) tutte le proprietà di tutti i depositi presenti nella tabella “mezzo”.

- importRow

```
Shared Function importRow(ByVal dr As DataRow) As
Depositi.Deposito
        Dim m As New Depositi.Deposito
        m.codice = dr.Item("codice").ToString
        m.smk = dr.Item("Tstart").ToString
        m.fmk = dr.Item("Tfinish").ToString
        Return m
    End Function
```

Associa i valori prelevati dall'i-esima riga della tabella “deposito” alla variabile m di tipo deposito.

- EsisteCodice

```
Public Shared Function EsisteCodice(ByVal codice As String) As
Boolean
        Dim cn As New connessione.connect
        Dim ds As New DataSet
        Dim str As String = ""
        Dim ce As Boolean = False
```

```
        str = "select codice from deposito where codice='" &
codice & "'"
        ds = cn.Leggidati(str)
        If ds.Tables(0).Rows.Count > 0 Then   ce = True
        End If
        ds.Dispose()
        Return ce
    End Function
```

Verifica l'esistenza di un codice specifico all'interno della tabella deposito.

- DeleteDeposito

```
Shared Function DeleteDeposito(ByVal codiceold As String) As
Boolean
        Dim cn As New connessione.connect
        Dim str As String = ""
        'Delete
        str = "Delete  from deposito "
        str &= " WHERE "
        str &= "Codice='" & codiceold & "'"
        cn.EseguiQueryDati(str)
        Return True
    End Function
```

Permette l'eliminazione dal database del deposito d'interesse.

La classe macello

Vengono definiti i seguenti attributi:

```
   Public codice As String
    Public s As String
    Public f As String
    Public pi As String
    Public gi As String
```

- Codice ➔ codice del macello
- S ➔ l'inizio della finestra temporale del macello
- F ➔ la fine della finestra temporale del macello
- pi ➔ carico fornito di pelle fresca
- gi ➔ tempo massimo entro il quale le pelli devono essere scaricate presso la conceria.

E i seguenti metodi:

- NuovoMacello

```
Public Function NuovoMacello(ByVal codice As String)
        Dim cn As New connessione.connect
        Dim str As String = ""

        'INSERT
        str = "insert into macello "
        str &=
"(Codice,PesoTotale,TempoMaxPelli,Tstart,Tfinish)"
        str &= "values"
        str &= "('" & codice & "','" & pi & "','" & gi & "','" &
s & "','" & f & "')"
        MsgBox(str)
        cn.EseguiQueryDati(str)
        Return True
        End Function
```

Inserisce nella tabella “macello” i valori degli attributi dell’oggetto considerato (chiave ➔ codice)

- SalvaMacello

```
Public Function SalvaMacello(ByVal codice As String) As Boolean
        Dim cn As New connessione.connect
        Dim str As String = ""

       'UPDATE
        str = "Update macello set "
        str &= "Codice='" & codice & "',"
        str &= "PesoTotale='" & pi & "',"
        str &= "TempoMaxPelli='" & gi & "',"
        str &= "Tstart='" & s & "',"
        str &= "Tfinish='" & f & "' "
        str &= " WHERE "
        str &= "Codice='" & codice & "'"

        cn.EseguiQueryDati(str)
        Return True
    End Function
```

Aggiorna i valori degli attributi di un macello già esistente nel database.

- caricaMacello

```
Public Shared Function CaricaMacello(ByVal codice As String) As
Macelli.macello
        Dim cn As New connessione.connect
        Dim ds As New DataSet
        Dim str As String = ""
        Dim m As Macelli.Macello
'seleziona dalla tabella Macello tutti i codici uguale a codice'
        str = "select * from Macello where codice='" & codice &
"'       ds = cn.Leggidati(str)
  se il numero di righe prelevato dal db è >0
        If ds.Tables(0).Rows.Count > 0 Then
 'metto nella variabile m l'intera riga
            m = Macello.importRow(ds.Tables(0).Rows(0))
        End If

        'chiusura connessione db
        ds.Dispose()
        Return m
    End Function
```

Restituisce i valori letti dalla tabella “macello” che hanno un determinato codice.

- CaricaMacelli

```
Public Shared Function CaricaMacelli() As List(Of
Macelli.Macello)
        Dim cn As New connessione.connect
        Dim ds As New DataSet
        Dim str As String = ""
        Dim i As Integer

        Dim Lm As New List(Of Macelli.Macello)      'crea una
variabile array
        'selezioniamo tutto dalla tabella macello
        str = "select * from macello "
        ds = cn.Leggidati(str)
        'controllo se il numero di righe prelevate dal db è >0
        If ds.Tables(0).Rows.Count > 0 Then
        'per ogni elemento prelevato dalla query lo inserisco
nell'array Lm
            For i = 0 To ds.Tables(0).Rows.Count - 1
                Lm.Add(importRow(ds.Tables(0).Rows(i)))
            Next
```

```
        End If
        ds.Dispose()
        Return Lm
    End Function
```

Carica all'interno di una lista tutti i macelli presenti nel database che presentano un determinato codice.

- CaricaMacelliDt

```
Public Shared Function CaricaMacelliDt() As DataTable
        Dim cn As New connessione.connect
        Dim ds As DataSet
        Dim dt As New DataTable
        Dim str As String = ""
        Dim i As Integer

        Dim Lm As New List(Of Macelli.Macello)      'crea una
variabile array
        'selezioniamo tutto dalla tabella mezzo
        str = "select * from macello "
        ds = cn.Leggidati(str)
        'controllo se il numero di righe prelevate dal db è > 0
        If ds.Tables(0).Rows.Count > 0 Then
            dt = ds.Tables(0)
        End If

        ds.Dispose()
        Return dt
    End Function '
```

Restituisce per mezzo di un datatable (memoria temporanea per la lettura di dati da un database) tutte le proprietà di tutti i macelli presenti nella tabella "mezzo".

- importRow

```
Shared Function importRow(ByVal dr As DataRow) As
Macelli.Macello
        Dim m As New Macelli.Macello

        m.codice = dr.Item("codice").ToString
        m.s = dr.Item("Tstart").ToString
        m.f = dr.Item("Tfinish").ToString
```

```
        m.pi = dr.Item("PesoTotale").ToString
        m.gi = dr.Item("TempoMaxPelli").ToString

        Return m
    End Function
```

Associa i valori prelevati dall'i-esima riga della tabella “macello” alla variabile m di tipo macello.

- EsisteCodice

```
Public Shared Function EsisteCodice(ByVal codice As String) As
Boolean
        Dim cn As New connessione.connect
        Dim ds As New DataSet
        Dim str As String = ""
        Dim ce As Boolean = False

str = "select codice from macello where codice='" & codice & "'"
        ds = cn.Leggidati(str)
        If ds.Tables(0).Rows.Count > 0 Then
            ce = True
        End If
                ds.Dispose()
        Return ce
    End Function
```

Verifica l’esistenza di un codice specifico all’interno della tabella deposito.

- DeleteMacello

```
Shared Function DeleteMacello(ByVal codiceold As String) As
Boolean
        Dim cn As New connessione.connect
        Dim str As String = ""
        'Delete
        str = "Delete  from macello "
        str &= " WHERE "
        str &= "Codice='" & codiceold & "'"
        cn.EseguiQueryDati(str)
        Return True
    End Function
```

Permette l’eliminazione dal database del macello d’interesse.

La classe gestione dati

La classe gestione dati è stata creata per gestire gli attributi corrispondenti alla coppia di oggetti macello – mezzo , e alla terna di oggetti macello – macello – mezzo.
Vengono infatti in essa definite due classi :

- La classe MacelloMezzo
- La classe MacelloMacelloMezzo

La classe **MacelloMezzo**

```
Public Class MacelloMezzo
      Structure Dati
        Public Macello As String
        Public Mezzo As String
        Public TempoCarico As String
        Public Tdepmacello As String
        Public Tmacelloconc As String
        Public Cdepmacello As String
        Public Cmacelloconc As String
        Public Costodicarico As String

        End Structure
```

- Macello ➔ codice del macello
- mezzo ➔ codice del mezzo

attributi relativi ad ogni coppia macello – mezzo:

- TempoCarico ➔ tempo di carico
- Tdepmacello ➔ tempo necessario al mezzo per raggiungere il macello partendo dal suo stesso deposito.
- Tmacelloconc ➔ tempo necessario al mezzo per raggiungere la conceria partendo dal macello.
- Cdepmacello ➔ costo del trasporto dal deposito al macello.
- Cmacelloconc ➔ costo del trasporto dal macello alla conceria
- Costodicarico ➔ costo del carico di pelle fresca che avviene presso il macello

La classe **MacelloMacelloMezzo**:

```
Public Class MacelloMAcelloMezzo
        Structure Dati
            Public MacelloA As String
            Public MacelloB As String
            Public Mezzo As String
            Public Tempo As String
            Public Costo As String

            End Structure
```

- MacelloA ➔ codice del macello A
- MacelloB ➔ codice del macello B
- Mezzo ➔ codice del mezzo

attributi relativi ad ogni terna macello – macello – mezzo :

- Tempo ➔ tempo necessario al mezzo per raggiungere il macello B partendo dal macello A.
- Costo ➔ costo del trasporto dal macello A al macello B.

I metodi definiti nella classe Gestione Dati sono:

- Carica salva Macello_Mezzo

```
Shared Sub SalvaMacello_Mezzo(ByVal datimm As Dati)
            Dim cn As New connessione.connect
            Dim ds As New DataSet
            Dim str As String = ""

            Dim Lm As New Dati 'crea una variabile array

            str = " delete from Macello_Mezzo WHERE macello='" &
datimm.Macello & "' and mezzo='" & datimm.Mezzo & "'"
            cn.EseguiQueryDati(str)

            'INSERT
            str = "insert into Macello_Mezzo "
            str &=
"(macello,mezzo,Tempodicarico,Tdepmacello,Tmacelloconc,Cdepmacel
lo,Cmacelloconc,Costodicarico)"
```

```
            str &= "values"
            str &= "('" & datimm.Macello & "','" & datimm.Mezzo
& "','" & datimm.TempoCarico & "','" & datimm.Tdepmacello &
"','" & datimm.Tmacelloconc & "','" & datimm.Cdepmacello & "','"
& datimm.Cmacelloconc & "','" & datimm.Costodicarico & "')"
            cn.EseguiQueryDati(str)
        End Sub
```

Questa funzione si connette al database alla tabella Macello_Mezzo, cancella gli attributi corrispondenti alla coppia macello – mezzo considerata e salva i nuovi valori inseriti.

- Carica Macello_Mezzo

```
Shared Function CaricaMacello_Mezzo(ByVal CodiceMacello As
String, ByVal codicemezzo As String) As Dati
            Dim cn As New connessione.connect
            Dim ds As New DataSet
            Dim str As String = ""
            Dim i As Integer

            Dim Lm As New Dati'crea una variabile array
            str = "select * from Macello_Mezzo WHERE macello='"
& CodiceMacello & "' and mezzo='" & codicemezzo & "'"
            ds = cn.Leggidati(str)
          'controllo se il numero di righe prelevate dal db è >0
            If ds.Tables(0).Rows.Count > 0 Then
                Lm = importRow(ds.Tables(0).Rows(0))
            End If
            ds.Dispose()
            Return Lm
        End Function
```

Carica all'interno di una lista tutti gli attributi corrispondenti alla coppia macello – mezzo considerata, selezionati dalla tabella Macello_mezzo.

- importRow

```
   Shared Function importRow(ByVal dr As DataRow) As Dati
            Dim m As New Dati

            m.Macello = dr.Item("Macello").ToString
            m.Mezzo = dr.Item("Mezzo").ToString
```

```
            m.TempoCarico = dr.Item("Tempodicarico").ToString
            m.Tdepmacello = dr.Item("Tdepmacello").ToString
            m.Tmacelloconc = dr.Item("Tmacelloconc").ToString
            m.Cdepmacello = dr.Item("Cdepmacello").ToString
            m.Cmacelloconc = dr.Item("Cmacelloconc").ToString
            m.Costodicarico = dr.Item("Costodicarico").ToString
            Return m
```

Associa i valori prelevati dall'i-esima riga della tabella macello – mezzo alla variabile m di tipo dati.

- Salva Macello_Mezzo

```
Shared Sub SalvaMacello_Mezzo(ByVal datimm As Dati)
            Dim cn As New connessione.connect
            Dim ds As New DataSet
            Dim str As String = ""
            Dim Lm As New Dati 'crea una variabile array
            str = " delete from Macello_Macello_Mezzo WHERE
macelloA='" & datimm.MacelloA & "' and macelloB='" &
datimm.MacelloB & "' and mezzo='" & datimm.Mezzo & "'"
            cn.EseguiQueryDati(str)
            'INSERT
            str = "insert into Macello_Macello_Mezzo "
            str &= "(macelloA,macelloB,mezzo,tempomm,costomm)"
            str &= "values"
            str &= "('" & datimm.MacelloA & "','" &
datimm.MacelloB & "','" & datimm.Mezzo & "','" & datimm.Tempo &
"','" & datimm.Costo & "')"
            cn.EseguiQueryDati(str)
        End Sub
```

Questa funzione si connette alla tabella Macello_Macello_Mezzo del database, cancella gli attributi corrispondenti alla terna macello – macello – mezzo considerata e salva i nuovi valori inseriti.

- Carica Macello_Mezzo

```
Shared Function CaricaMacello_Macello_Mezzo(ByVal CodiceMacelloA
As String, ByVal CodiceMacelloB As String, ByVal codicemezzo As
String) As Dati
            Dim cn As New connessione.connect
            Dim ds As New DataSet
```

```
            Dim str As String = ""
            Dim i As Integer

            Dim Lm As New Dati 'crea una variabile array
            str = "select * from Macello_Macello_Mezzo WHERE
macelloa='" & CodiceMacelloA & "' and macellob='" &
CodiceMacelloB & "' and mezzo='" & codicemezzo & "'"
            ds = cn.Leggidati(str)
          'controllo se il numero di righe prelevate dal db è >0
            If ds.Tables(0).Rows.Count > 0 Then
                Lm = importRow(ds.Tables(0).Rows(0))
            End If
            ds.Dispose()
            Return Lm
        End Function
```

Carica all'interno di una lista tutti gli attributi corrispondenti alla terna macello – macello – mezzo considerata, selezionati dalla tabella macello – macello – mezzo

.

- importRow

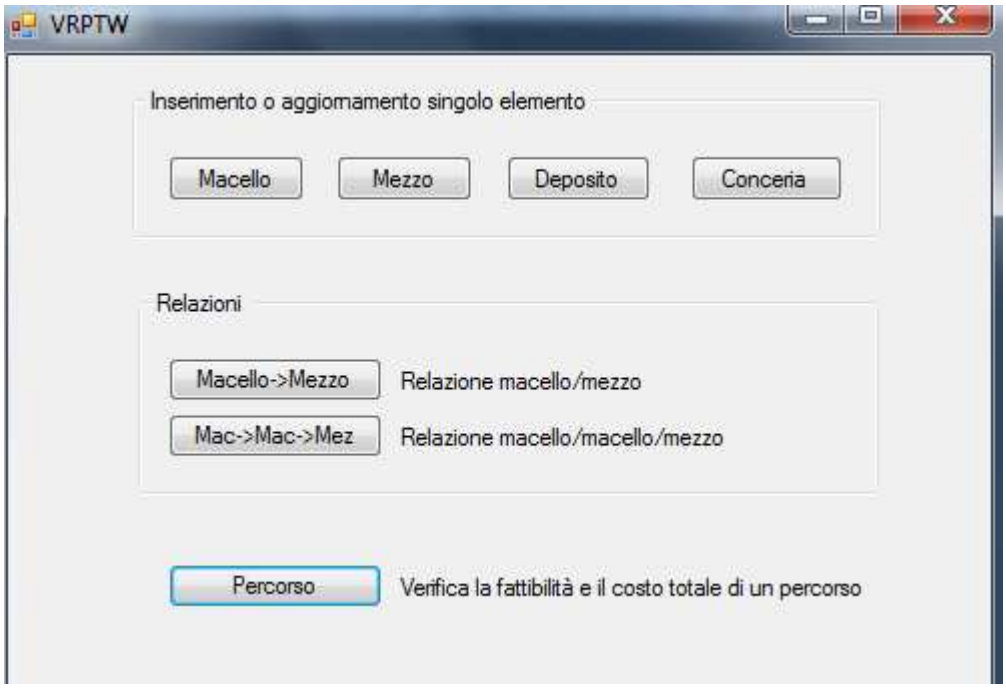
```
    Shared Function importRow(ByVal dr As DataRow) As Dati
            Dim m As New Dati

            m.MacelloA = dr.Item("Macelloa").ToString
            m.MacelloB = dr.Item("Macellob").ToString
            m.Mezzo = dr.Item("Mezzo").ToString
            m.Tempo = dr.Item("TempoMM").ToString
            m.Costo = dr.Item("CostoMM").ToString

            Return m
        End Function
```

Associa i valori prelevati dall'i-esima riga della tabella Macello_Macello_Mezzo alla variabile m di tipo dati.

3.2 Applicazioni Windows Form

Le applicazioni Windows Form che ho trovato necessario creare sono:

- l' applicazione principale **VRPTW**

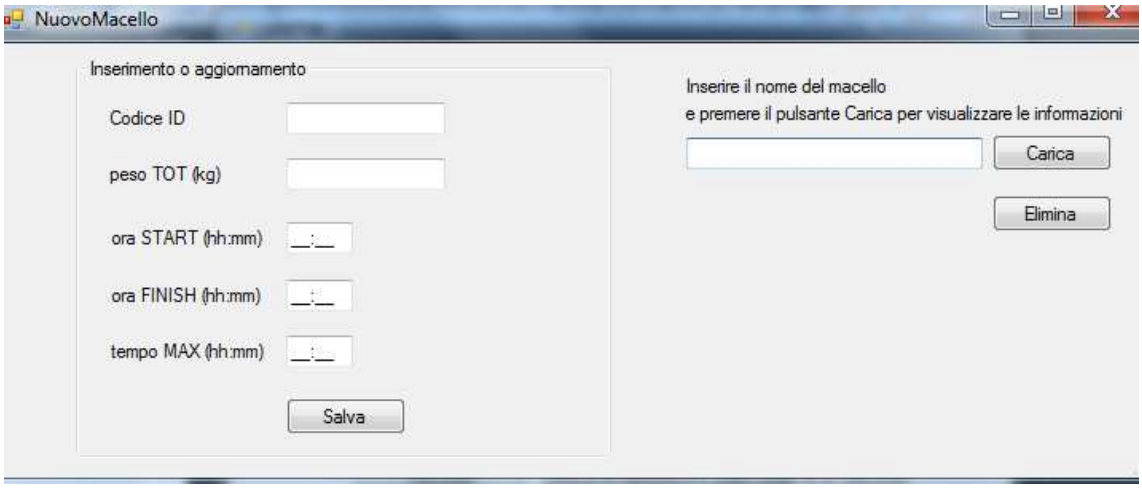


L'interfaccia principale la quale, tramite i vari "pulsanti" consente il collegamento alle applicazioni:

- l' applicazione **NuovoMacello** (pulsante "Macello")
- l' applicazione **NuovoMezzo** (pulsante "Mezzo")
- l' applicazione **NuovoDeposito** (pulsante "Deposito")
- l'applicazione **AggiornaConceria** (pulsante "Conceria")
- l' applicazione **Macello-Mezzo** (pulsante "Macello->Mezzo")
- l' applicazione **Macello-Macello_Mezzo** (pulsante "Macello->Macello->Mezzo")
- l' applicazione **percorso** (pulsante "Percorso")

Applicazione NuovoMacello

Consente l'inserimento nel database (tabella "macello") di un nuovo oggetto di tipo "macello" e di tutte le informazioni relative ai corrispondenti attributi (obbligatorie), l'aggiornamento delle informazioni relative ad un macello già presente nel database, o la semplice visualizzazione delle informazioni del macello selezionato.



- il pulsante "Salva"

assegna a delle variabili temporanee i valori inseriti dall'utente presso l'interfaccia:

```
        Dim codiceid = Me.txtCodice.Text
        Dim tempoS = Me.txtoraS.Text
        Dim tempoF = Me.txtoraF.Text
        Dim Ptot = Me.txtPeso.Text
        Dim Ttot = Me.txtTmax.Text
```

verifica che il nuovo macello inserito non sia già esistente all'interno del database:

```
        If Macello.EsisteCodice(codiceid) And Me.lbverifica.Text
= "Nuovo"  Then
            Exit Sub
        End If
```

esegue una serie di controlli per verificare che vengano inserite le informazioni relative a tutti gli attributi del nuovo macello.

In caso un valore non venga inserito, comparirà la scritta “inserire valore” a fianco al TextBox corrispondente:

```
If codiceid = "" Then Me.lbCodice.Visible = True
    Exit Sub
    End If

If tempoS = "" Then Me.lbOraS.Visible = True
    Exit Sub
    End If

If tempoF = "" Then  Me.lbOraF.Visible = True
    Exit Sub
    End If

If Ptot = "" Then Me.lbpeso.Visible = True
    Exit Sub
    End If

If Ttot = "" Then Me.lbTmax.Visible = True
    Exit Sub
    End If
```

crea un nuovo oggetto di tipo “macello” e assegna i valori inseriti dall’utente ai rispettivi attributi:

```
Dim ma As New Macelli.Macello
ma.codice = codiceid
ma.s = tempoS
ma.f = tempoF
ma.pi = Ptot
ma.gi = Ttot
```

richiama il metodo NuovoMacello o il metodo SalvaMacello a seconda che si tratti dell’inserimento di un nuovo macello o dell’aggiornamento di un macello già presente nel database:

```
Dim isok As Boolean
If Me.lbverifica.Text = "Nuovo" Then
    isok = ma.NuovoMacello(codiceid)
    Me.lbverifica.Text = codiceid
    Else
    isok = ma.SalvaMacello(codiceid)
End If
```

informa l'utente della riuscita del salvataggio o eventualmente dela sua non riuscita:

```
        If isok Then
            Me.lEsitoOperazione.Text = "salvataggio ok"
        Else
            Me.lEsitoOperazione.Text = "errore.."
        End If
```

- il pulsante "carica"

ricerca nel database il macello che presenta il codice inserito dall'utente, rendendo visibili i valori di tutti i suoi attributi.

```
Dim m As New Macello
        m = Macello.CaricaMacello(Me.txtcaricaMac.Text)
        If Not IsNothing(m) Then
            Me.txtCodice.Text = m.codice
            Me.txtoraF.Text = m.f
            Me.txtoraS.Text = m.s
            Me.txtPeso.Text = m.pi
            Me.txtTmax.Text = m.gi

            Me.lbverifica.Text = m.codice

        End If
```

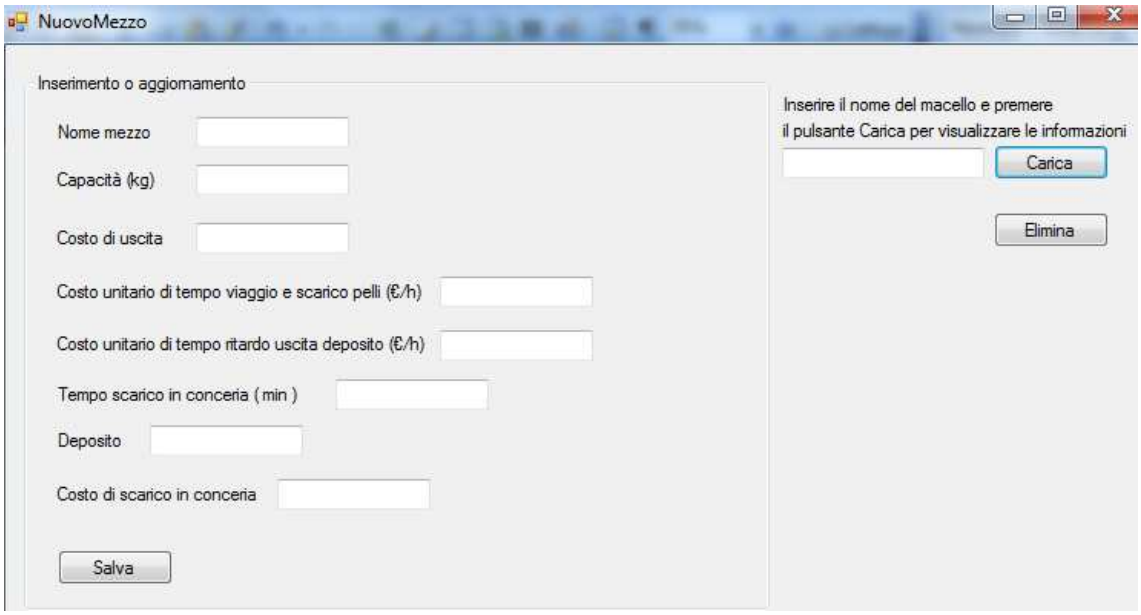
- il pulsante "elimina"

ricerca nel database il macello che presenta il codice inserito dall'utente, e lo elimina utilizzando la funzione DeleteMacello..

```
Macello.DeleteMacello(Me.txtcaricaMac.Text)
        Me.lbldelete.Visible = True
        Me.lbldelete.Text = Me.txtcaricaMac.Text & " cancellato"
```

Applicazione NuovoMezzo

Consente l'inserimento nel database (tabella "mezzo") di un nuovo oggetto di tipo "mezzo" e di tutte le informazioni relative ai corrispondenti attributi (obbligatorie), l'aggiornamento delle informazioni relative ad un mezzo già presente nel database, o la semplice visualizzazione delle informazioni del mezzo selezionato.



- il pulsante "Salva"

assegna a delle variabili temporanee i valori inseriti dall'utente presso l'interfaccia:

```
        Dim nome As String = Me.txtNome.Text
        Dim cap As String = Me.txtCapacita.Text
        Dim costk As String = Me.txtCostoUscita.Text
        Dim ctk As String = Me.txtctk.Text
        Dim cdk As String = Me.txtcdk.Text
        Dim tk As String = Me.txttk.Text
        Dim dep As String = Me.txtdeposito.Text
        Dim cl As String = Me.txtcostoscarico.Text
```

verifica che il nuovo mezzo inserito non sia già esistente all'interno del database:

```
        If Mezzo.EsisteCodice(nome) And Me.lbverifica.Text =
"Nuovo" Then Exit Sub
        End If
```

esegue una serie di controlli per verificare che vengano inserite le informazioni relative a tutti gli attributi del nuovo mezzo.
In caso un valore non venga inserito, comparirà la scritta “inserire valore” a fianco al TextBox corrispondente:

```
If nome = "" Then Me.Label4.Visible = True
Exit Sub
End If

If cap = "" Then Me.Label10.Visible = True
Exit Sub
End If

If costk = "" Then Me.Label11.Visible = True
Exit Sub
End If

If ctk = "" Then Me.Label12.Visible = True
Exit Sub
End If

If cdk = "" Then Me.Label13.Visible = True
Exit Sub
End If

If tk = "" Then Me.Label14.Visible = True
Exit Sub
End If

If dep = "" Then Me.Label15.Visible = True
Exit Sub
End If

If cl = "" Then Me.Label18.Visible = True
Exit Sub
End If
```

crea un nuovo oggetto di tipo “mezzo” e assegna i valori inseriti dall’utente ai rispettivi attributi:

```
Dim m As New mezzo
m.nome = nome
m.ck = costk
m.qk = cap
```

```
m.ctk = ctk
m.cdk = cdk
m.tk = tk
m.dep = dep
m.cl = cl
```

richiama il metodo NuovoMezzo o il metodo SalvaMezzo a seconda che si tratti dell'inserimento di un nuovo mezzo o dell'aggiornamento di un mezzo già presente nel database:

```
Dim isok As Boolean
If Me.lbverifica.Text = "Nuovo" Then
    isok = m.NuovoMezzo(nome)
Else
    isok = m.SalvaMezzo(Me.lbCodice.Text)
End If
```

informa l'utente della riuscita del salvataggio o eventualmente dela sua non riuscita:

```
If isok Then
    Me.Label5.Text = "salvataggio ok"
Else
    Me.Label5.Text = "errore.."
End If
```

ricerca nel database il mezzo che presenta il codice inserito dall'utente, rendendo visibili i valori di tutti i suoi attributi.

- Il pulsante "Carica"

```
Dim m As New Mezzi.Mezzo
        m = Mezzo.CaricaMezzo(Me.TextBox1.Text)
        If Not IsNothing(m) Then
            Me.txtCapacita.Text = m.qk
            Me.txtCostoUscita.Text = m.ck
            Me.txtNome.Text = m.nome
            Me.lbCodice.Text = m.nome
            Me.txtctk.Text = m.ctk
            Me.txtcdk.Text = m.cdk
            Me.txttk.Text = m.tk
            Me.txtdeposito.Text = m.dep
            Me.txtcostoscarico.Text = m.cl
```

```
            Me.lbverifica.Text = "Modifica"
        End If
```

- il pulsante “elimina”

ricerca nel database il mezzo che presenta il codice inserito dall’utente, e lo elimina utilizzando la funzione DeleteMezzo.

```
Mezzo.DeleteMezzo(Me.TextBox1.Text)
        Me.lbcanc.Visible = True
```

Applicazione NuovoDeposito

Analogamente alle altre due applicazioni NuovoMacello e NuovoMezzo, l’applicazione NuovoDeposito consente l’inserimento nel database, l’aggiornamento e la visualizzazione all’interfaccia di tutte le informazioni relative ad un oggetto “deposito”.

- il pulsante “Salva”

assegna a delle variabili temporanee i valori inseriti dall’utente presso l’interfaccia:

```
        Dim codice As String = Me.txtCodice.Text
        Dim smk As String = Me.txtoraS.Text
        Dim fmk As String = Me.txtoraF.Text
```

verifica che il nuovo deposito inserito non sia già esistente all'interno del database (tabella "deposito"):

```
        If Deposito.EsisteCodice(codice) And Me.lbverifica.Text
= "Nuovo" Then
            Exit Sub
        End I
```

esegue una serie di controlli per verificare che vengano inserite le informazioni relative a tutti gli attributi del nuovo deposito.
In caso un valore non venga inserito, comparirà la scritta "inserire valore" a fianco al TextBox corrispondente:

```
        If codice = "" Then Me.lbCodice.Visible = True
            Exit Sub
        End If

        If smk = "" Then Me.lbS.Visible = True
            Exit Sub
        End If

        If fmk = "" Then Me.lbF.Visible = True
            Exit Sub
        End If
```

crea un nuovo oggetto di tipo "deposito" e assegna i valori inseriti dall'utente ai rispettivi attributi:

```
        Dim d As New Deposito
        d.codice = codice
        d.smk = smk
        d.fmk = fmk
```

richiama il metodo NuovoDeposito o il metodo SalvaDeposito a seconda che si tratti dell'inserimento di un nuovo deposito o dell'aggiornamento di un deposito già presente nel database:

```
        Dim isok As Boolean
        If Me.lbverifica.Text = "Nuovo" Then
            isok = d.NuovoDeposito(codice)
        Else
            isok = d.SalvaDeposito(Me.txtCodice.Text)
        End If
```

informa l'utente della riuscita del salvataggio o eventualmente dela sua non riuscita:

```
        If isok Then
            Me.Label2.Visible = True
        Else
            Me.Label2.Text = "errore.."
            Me.Label2.Visible = True
        End If
```

- il pulsante "Carica"

ricerca nel database il mezzo che presenta il codice inserito dall'utente, rendendo visibili alll'utente i valori di tutti i suoi attributi.

```
Dim d As New Depositi.Deposito
        d = Deposito.CaricaDeposito(Me.txtcaricaDep.Text)
        If Not IsNothing(d) Then
            Me.txtCodice.Text = d.codice
            Me.txtoraS.Text = d.smk

            Me.txtoraF.Text = d.fmk
            Me.lbverifica.Text = "Modifica"
        End If
```

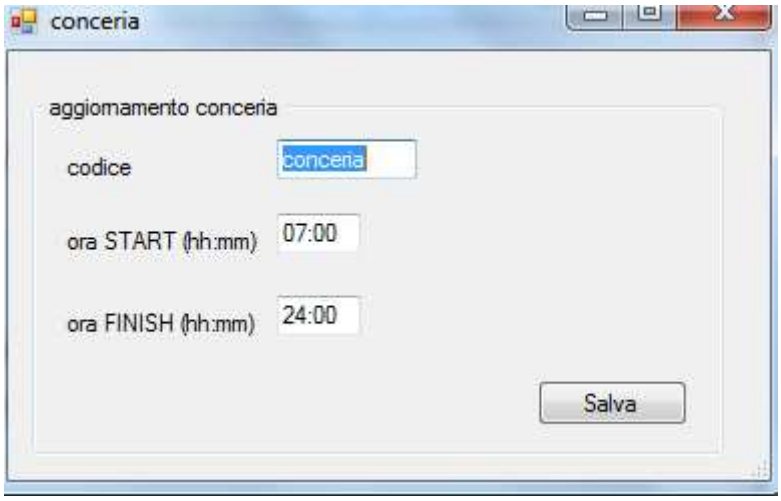
- il pulsante "elimina"

ricerca nel database il deposito che presenta il codice inserito dall'utente, e lo elimina utilizzando la funzione DeleteDeposito.

```
Deposito.DeleteDeposito(Me.txtcaricaDep.Text)
        Me.Lbcanc.Visible = True
```

Applicazione AggiornaConceria

L'applicazione AggiornaConceria consente l'aggiornamento relativo alle informazioni sulla finestra temporale della conceria.



- il pulsante "Salva"

assegna a delle variabili temporanee i valori inseriti dall'utente presso l'interfaccia:

```
Dim codiceid = Me.txtCodice.Text
Dim tempoS = Me.txtoraS.Text
Dim tempoF = Me.txtoraF.Text
```

esegue una serie di controlli per verificare che vengano inserite tutte le informazioni. In caso un valore non venga inserito, comparirà la scritta "inserire valore" a fianco al TextBox corrispondente:

```
If codiceid = "" Then
    Me.Lblvercod.Visible = True
    Exit Sub
End If

If tempoS = "" Then
    Me.LblveroraS.Visible = True
    Exit Sub
End If

If tempoF = "" Then
```

```
        Me.lbveroraF.Visible = True
        Exit Sub
    End If
```

crea un nuovo oggetto di tipo “conceria” e assegna i valori inseriti dall’utente ai rispettivi attributi:

```
    Dim con As New Conceria.Conceria
    con.codice = codiceid
    con.s = tempoS
    con.f = tempoF
```

richiama il metodo SalvaConceria per effettuare l’aggiornamento delle informazioni.

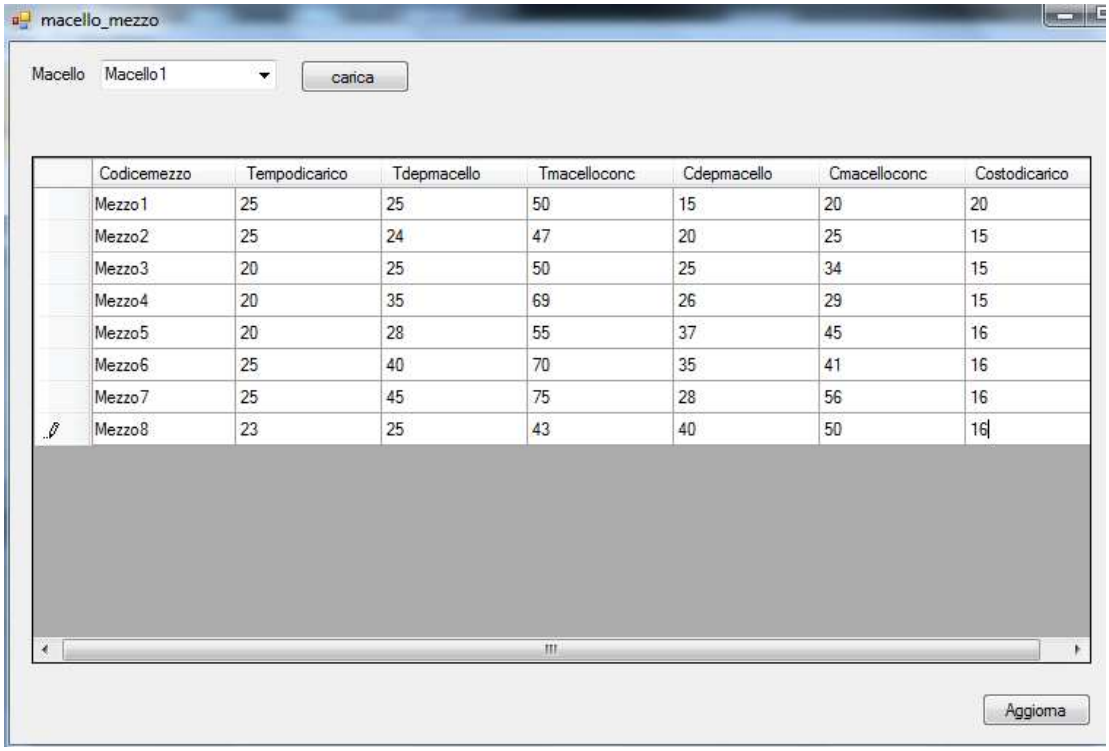
```
    Dim isok As Boolean
    isok = con.SalvaConceria(codiceid)
```

informa l’utente della riuscita del salvataggio o eventualmente dela sua non riuscita:

```
    If isok Then
        Me.Labelverifica.Visible = True
    Else
        Me.Labelverifica.Text = "errore.."
        Me.Labelverifica.Visible = True
    End If
```

Applicazione Macello-Mezzo

Consente l'aggiornamento e salvataggio nel database (tabella "Macello_Mezzo"), la modifica o la semplice visualizzazione di un oggetto di tipo "MacelloMezzo", definito dalla classe "GestioneDati" i cui attributi corrispondono a tutte le informazioni relative alla coppia di oggetti macello – mezzo, che coincidono con i campi della tabella.



La selezione del macello avviene grazie alla funzione

```
Protected Sub CaricaMacelli()
        Dim Lm As List(Of Macello)
        Dim m As Macello
        Dim i As Integer
        Lm = Macello.CaricaMacelli

        For i = 0 To Lm.Count - 1
            listaMacelli.Items.Add(Lm(i).codice)
        Next
```

Che ricorre al metodo "Carica Macelli" per creare una lista da cui l'utente possa selezionare il macello di proprio interesse.

- il pulsante “Carica”

crea I campi della tabella, I quali vengono fatti corrispondere agli attributi dell’oggetto MacelloMezzo, associati, riga per riga, al macello selezionato e al mezzo corrispondente nella prima colonna:

```
Dim Lme As IList(Of Mezzo)
        Dim Lmm As New
GestioneDati.GestioneDati.MacelloMezzo.Dati

        Lme = Mezzo.CaricaMezzi
        Dim i As Integer
        Dim smezzo As String
        Dim smacello As String = listaMacelli.SelectedItem
        Dim dt As New DataTable
        Dim dc As DataColumn

        dc = New DataColumn("Codicemezzo")
        dc.DataType = System.Type.GetType("System.String")
        dt.Columns.Add(dc)

        dc = New DataColumn("Tempodicarico")
        dc.DataType = System.Type.GetType("System.String")
        dt.Columns.Add(dc)

        dc = New DataColumn("Tdepmacello")
        dc.DataType = System.Type.GetType("System.String")
        dt.Columns.Add(dc)

        dc = New DataColumn("Tmacelloconc")
        dc.DataType = System.Type.GetType("System.String")
        dt.Columns.Add(dc)

        dc = New DataColumn("Cdepmacello")
        dc.DataType = System.Type.GetType("System.String")
        dt.Columns.Add(dc)

        dc = New DataColumn("Cmacelloconc")
        dc.DataType = System.Type.GetType("System.String")
        dt.Columns.Add(dc)

        dc = New DataColumn("Costodicarico")
        dc.DataType = System.Type.GetType("System.String")
        dt.Columns.Add(dc)
```

per ogni mezzo presente nel database, rende visibili all'utente i valori degli attributi dell'oggetto MacelloMezzo (il macello selezionato rimane costante):

```
Dim dr As DataRow
        For i = 0 To Lme.Count - 1
            smezzo = Lme.Item(i).nome
            Lmm =
GestioneDati.GestioneDati.MacelloMezzo.CaricaMacello_Mezzo(smace
llo, smezzo)

            dr = dt.NewRow()
            dr.Item("Codicemezzo") = smezzo
            dr.Item("Tdepmacello") = Lmm.Tdepmacello
            dr.Item("Tmacelloconc") = Lmm.Tmacelloconc
            dr.Item("Cdepmacello") = Lmm.Cdepmacello
            dr.Item("Cmacelloconc") = Lmm.Cmacelloconc
            dr.Item("Costodicarico") = Lmm.Costodicarico
            dr.Item("Tempodicarico") = Lmm.TempoCarico
            dt.Rows.Add(dr)
```

- pulsante "salva"

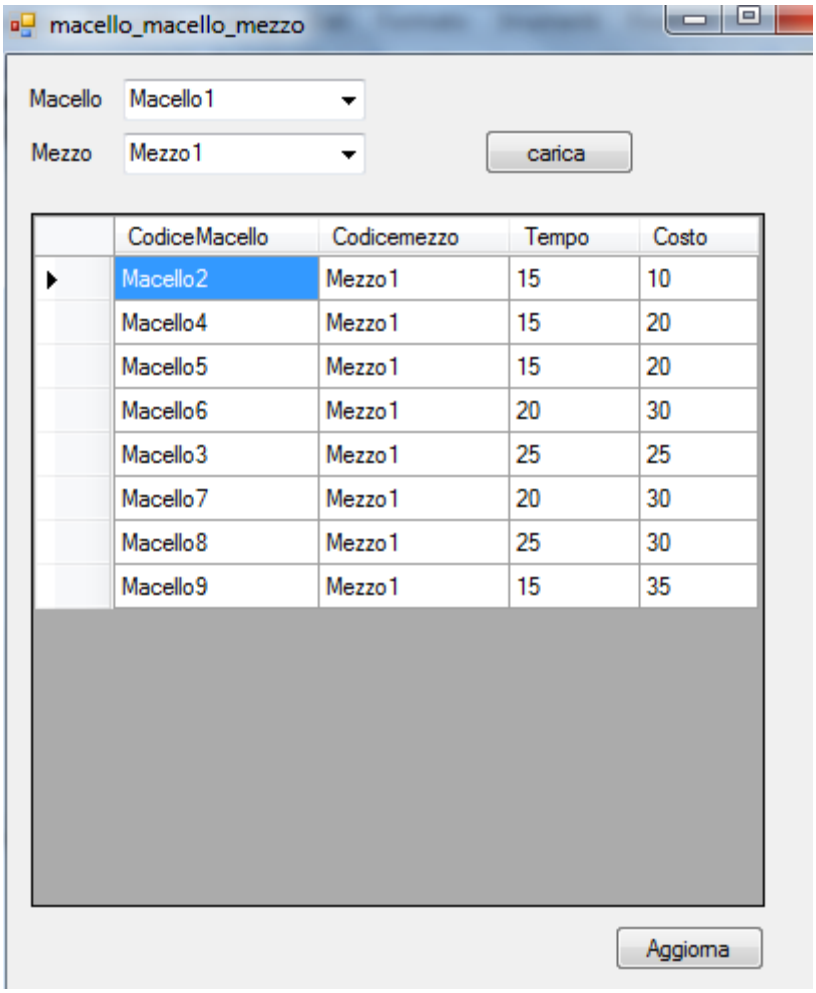
assegna i valori inseriti dall'utente agli attributi dell'oggetto MacelloMezzo corrispondente nel database.

```
Dim smacello As String = listaMacelli.SelectedItem
        Dim i, k As Integer
        k = DataGridView1.RowCount
        Dim DatiMM As New
GestioneDati.GestioneDati.MacelloMezzo.Dati
        For i = 0 To k - 1
            DatiMM.Macello = smacello
            DatiMM.Mezzo =
DataGridView1.Rows(i).Cells(0).Value.ToString
            DatiMM.TempoCarico =
DataGridView1.Rows(i).Cells(1).Value.ToString
            DatiMM.Tdepmacello =
DataGridView1.Rows(i).Cells(2).Value.ToString
            DatiMM.Tmacelloconc =
DataGridView1.Rows(i).Cells(3).Value.ToString
            DatiMM.Cdepmacello =
DataGridView1.Rows(i).Cells(4).Value.ToString
            DatiMM.Cmacelloconc =
DataGridView1.Rows(i).Cells(5).Value.ToString
```

```
            DatiMM.Costodicarico =
DataGridView1.Rows(i).Cells(6).Value.ToString


GestioneDati.GestioneDati.MacelloMezzo.SalvaMacello_Mezzo(DatiMM
)
```

Applicazione Macello-Macello-Mezzo

Consente l'aggiornamento e il salvataggio nel database (tabella "Macello_Macello-Mezzo"), la modifica o la semplice visualizzazione di un oggetto di tipo "MacelloMacelloMezzo", definito dalla classe "GestioneDati" i cui attributi corrispondono a tutte le informazioni relative alla terna di oggetti Macello_Macello-Mezzo, che coincidono con i campi della tabella.



La selezione del macello avviene grazie alla funzione

```
Protected Sub CaricaMacelli()
        Dim Lm As List(Of Macello)
        Dim m As Macello
        Dim i As Integer
        Lm = Macello.CaricaMacelli
        For i = 0 To Lm.Count - 1
            listaMacelli.Items.Add(Lm(i).codice)
        Next
```

La selezione del mezzo avviene grazie alla funzione

```
Protected Sub CaricaMezzi()
        Dim Lm As List(Of Mezzo)
        Dim m As Mezzo
        Dim i As Integer
        Lm = Mezzo.CaricaMezzi
        listaMezzi.Items.Add("")
        For i = 0 To Lm.Count - 1
            listaMezzi.Items.Add(Lm(i).nome)
        Next
```

Che ricorrono ai metodo “CaricaMacelli” e “CaricaMezzi” per creare una lista da cui l’utente possa selezionare il macello e il mezzo di proprio interesse.

- il pulsante “Carica”

crea I campi della tabella, I quali vengono fatti corrispondere agli attributi dell’oggetto MacelloMacelloMezzo, associati, riga per riga, al macello e mezzo selezionato e al macello corrispondente nella prima colonna:

```
Dim Lmm As New
GestioneDati.GestioneDati.MacelloMAcelloMezzo.Dati

        'elenco macelli
        Dim Lmac As List(Of Macello)
        Lmac = Macello.CaricaMacelli

        'elenco mezzi
        Dim Lme As IList(Of Mezzo)
        Lme = Mezzo.CaricaMezzi(Me.listaMezzi.SelectedItem)
```

```
        Dim i, k As Integer
        Dim smezzo As String
        Dim smacelloA As String = listaMacelli.SelectedItem
        Dim smacelloB As String

        Dim dt As New DataTable
        Dim dc As DataColumn
        dc = New DataColumn("CodiceMacello")
        dc.DataType = System.Type.GetType("System.String")
        dt.Columns.Add(dc)

        dc = New DataColumn("Codicemezzo")
        dc.DataType = System.Type.GetType("System.String")
        dt.Columns.Add(dc)

        dc = New DataColumn("Tempo")
        dc.DataType = System.Type.GetType("System.String")
        dt.Columns.Add(dc)

        dc = New DataColumn("Costo")
        dc.DataType = System.Type.GetType("System.String")
        dt.Columns.Add(dc)
```

per ogni macello presente nel database, rende visibili all'utente i valori degli attributi dell'oggetto MacelloMacelloMezzo (il macello e mezzo selezionati rimangono fissi):

```
Dim dr As DataRow
        For k = 0 To Lmac.Count - 1
            smacelloB = Lmac(k).codice
            If smacelloB <> smacelloA Then
                For i = 0 To Lme.Count - 1
                    smezzo = Lme.Item(i).nome

 Lmm =
GestioneDati.GestioneDati.MacelloMAcelloMezzo.CaricaMacello_Mace
llo_Mezzo(smacelloA, smacelloB, smezzo)
                    dr = dt.NewRow()
                    dr.Item("Codicemezzo") = smezzo
                    dr.Item("CodiceMacello") = smacelloB
                    dr.Item("costo") = Lmm.Costo
                    dr.Item("tempo") = Lmm.Tempo

                    dt.Rows.Add(dr)
                Next
            End If
```

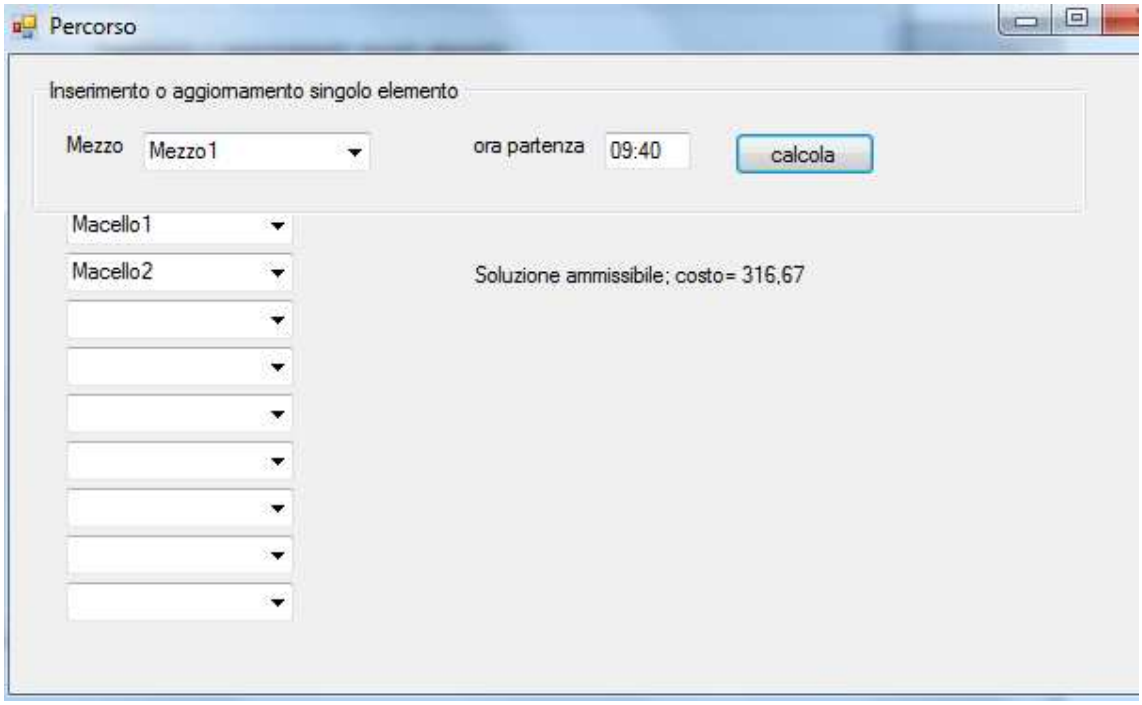
- pulsante “salva”

assegna i valori inseriti dall’utente agli attributi dell’oggetto MacelloMacelloMezzo corrispondente nel database.

```
Dim smacello As String = listaMacelli.SelectedItem
        Dim i, k As Integer
        k = DataGridView1.RowCount
        Dim DatiMM As New
GestioneDati.GestioneDati.MacelloMAcelloMezzo.Dati
        For i = 0 To k - 1
            DatiMM.MacelloA = smacello
            DatiMM.MacelloB =
DataGridView1.Rows(i).Cells(0).Value
            DatiMM.Mezzo = DataGridView1.Rows(i).Cells(1).Value
            DatiMM.Tempo =
DataGridView1.Rows(i).Cells(2).Value.ToString
            DatiMM.Costo =
DataGridView1.Rows(i).Cells(3).Value.ToString

GestioneDati.GestioneDati.MacelloMAcelloMezzo.SalvaMacello_Mezzo
(DatiMM)
```

Applicazione Percorso

E' il cuore del programma. Dato un mezzo ed un percorso selezionato dall'utente, questa applicazione verifica che tutti i vincoli di capacità e finestre temporali siano rispettati, e, in caso positivo, calcola il costo totale del percorso.



Per poter definire il percorso vengono create tante liste di selezione di macelli quanti sono i macelli presenti nel database:

```
Dim x As ComboBox
        Dim namecombo As String
        For i = 0 To Lm.Count - 1
            namecombo = "combo" & i
            x = New ComboBox
            x.Name = namecombo
            x.Location = New Point(30, 80 + 25 * i)
            x.Items.Add("")
  For j = 0 To Lm.Count – 1  x.Items.Add(Lm(j).codice)
            Next
            Me.Controls.Add(x)
        Next
    End Sub
    Private Sub Button1_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles Button1.Click
```

```
        risposta.Text = ""
```

La selezione del mezzo avviene grazie alla funzione:

```
Protected Sub CaricaMezzi()
        Dim Lm As List(Of Mezzo)
        Dim m As Mezzo
        Dim i As Integer
        Lm = Mezzo.CaricaMezzi
        listaMezzi.Items.Add("")
        Dim lss As New List(Of String)
        For i = 0 To Lm.Count - 1
            listaMezzi.Items.Add(Lm(i).nome)
        Next
```

La selezione del macello avviene grazie alla funzione:

```
    Protected Sub caricaMacelli()
        Dim Lm As List(Of Macello)
        Dim i, j As Integer
        Lm = Macello.CaricaMacelli
        Dim lss As New List(Of String)
        lss.Add("")
        For i = 0 To Lm.Count - 1
            lss.Add(Lm(i).codice)
        Next
```

che anche qui ricorrono ai metodo “CaricaMezzi” e “CaricaMacelli” per creare una lista da cui l’utente possa selezionare il mezzo e i macelli (percorso) di proprio interesse.

- pulsante “calcola”

definisce una serie di variabili necessarie:

```
        Dim costo_tot As Double = 0
        Dim a As TimeSpan =
CDate("12:30").Subtract(CDate("8:00"))
        Dim iM As Integer = Macello.CaricaMacelli.Count
        Dim lme As New Dictionary(Of String, String)
        Dim lcombo As New Dictionary(Of String, String)
        Dim x As ComboBox
```

```
        Dim namecombo As String
        Dim caricoattuale As Decimal = 0
        Dim caricomace As Decimal = 0
        Dim tmik As String ' tempo
        Dim si As String ' finstra mac
        Dim fi As String ' fistra mac
        Dim ss As String = Me.MaskedTextBox1.Text 'ora partetnza
        Dim tempodicarico As String
        Dim macelloprecedente As String
        Dim ORAATTUALE As Date
        Dim dSS1 As Date
```

controlla che sia stato selezionato un mezzo e in caso contrario avverte l'utente:

```
        If Me.listaMezzi.SelectedItem = "" Then
            risposta.Text = "Selezionare un mezzo" &
a.TotalMinutes
            Exit Sub
        End If
```

Verifica la singolarità di ogni macello presente nel percorso e in caso contrario avverte l'utente:

```
For i = 0 To iM - 1
            namecombo = "combo" & i
            x = CType(Controls(namecombo), ComboBox)
            If x.SelectedItem <> "" Then
                If Not lme.ContainsKey(x.SelectedItem) Then
                    lcombo.Add(x.SelectedItem, namecombo)
                    lme.Add(x.SelectedItem, x.SelectedItem)
                Else
                risposta.Text = "Macello selezionato piu volte"
                    Exit Sub
                End If
            End If
```

Assegna a delle variabili I valori degli attribute del mezzo selezionato:

```
        Dim deposito As String =
Mezzo.CaricaMezzo(listaMezzi.SelectedItem).dep
        Dim ck As String =
Mezzo.CaricaMezzo(listaMezzi.SelectedItem).ck
```

```
        Dim cdk As String =
Mezzo.CaricaMezzo(listaMezzi.SelectedItem).cdk
        Dim ctk As String =
Mezzo.CaricaMezzo(listaMezzi.SelectedItem).ctk
        Dim tk As String =
Mezzo.CaricaMezzo(listaMezzi.SelectedItem).tk
        Dim c_scarconc As String =
Mezzo.CaricaMezzo(listaMezzi.SelectedItem).cl
        Dim depo As Depositi.Deposito =
Depositi.Deposito.CaricaDeposito(deposito)
        Dim smk As String = depo.smk
        Dim fmk As String = depo.fmk    '
```

Verifica la finestra temporale del deposito:

```
        If Not (ss >= smk And ss <= fmk) Then
          risposta.Text = "finestra temporale errata: deposito"
            Exit Sub
        End If
```

Verifica che vengano rispettate le finestre temporali dei macelli e il vincolo di carico massimo del mezzo:

```
iM = lme.Count
Dim dRit As TimeSpan = New TimeSpan(0)

For i = 0 To lme.Count - 1

namecombo = lcombo(lme.Keys(i))
x = CType(Controls(namecombo), ComboBox)
If x.SelectedItem <> "" Then
  If IsNumeric(Macello.CaricaMacello(x.SelectedItem).pi) Then
          caricomace = Macello.CaricaMacello(x.SelectedItem).pi
                Else
                    caricomace = 0
                End If
                caricoattuale = caricoattuale + caricomace
'primo macello
If i = 0 Then
Dim macmez As GestioneDati.GestioneDati.MacelloMezzo.Dati =
GestioneDati.GestioneDati.MacelloMezzo.CaricaMacello_Mezzo(x.Sel
ectedItem, listaMezzi.SelectedItem)
              tmik = macmez.Tdepmacello
              tempodicarico = macmez.TempoCarico
              Dim xmacello As Macello
```

```
                xmacello = Macello.CaricaMacello(x.SelectedItem)

                      si = xmacello.s
                      fi = xmacello.f

                      Dim dSS As Date = CDate(ss)
                      Dim dsi As Date = CDate(si)
                      Dim dfi As Date = CDate(fi)

                      dSS = dSS.AddMinutes(tmik)

                      If (dSS <= dsi) Then
                      dRit = dsi.Subtract(dSS)
                      dSS = dsi
              risposta.Text = "aggiunto costo di ritardo d'uscita
"                      End If

If Not (dSS.AddMinutes(tempodicarico) <= dfi)  Then
risposta.Text = "finestra temporale errata: macello:" &
xmacello.codice
                      Exit Sub
                      End If

dSS = dSS.AddMinutes(tempodicarico)
ORAATTUALE = dSS
End If

If i > 0 And i < iM Then
Dim macmacmez As
GestioneDati.GestioneDati.MacelloMAcelloMezzo.Dati =
GestioneDati.GestioneDati.MacelloMAcelloMezzo.CaricaMacello_Mace
llo_Mezzo(macelloprecedente, x.SelectedItem,
listaMezzi.SelectedItem)
Dim tempodiviaggio As String = macmacmez.Tempo
Dim dSS As Date
dSS = ORAATTUALE
dSS = dSS.AddMinutes(tempodiviaggio)
Dim xmacello As Macello
xmacello = Macello.CaricaMacello(x.SelectedItem)

Dim macmez As GestioneDati.GestioneDati.MacelloMezzo.Dati =
GestioneDati.GestioneDati.MacelloMezzo.CaricaMacello_Mezzo(x.Sel
ectedItem, listaMezzi.SelectedItem)
mik = macmez.Tdepmacello
tempodicarico = macmez.TempoCarico
Dim tmaccon As String = macmez.Tmacelloconc
                      si = xmacello.s
```

```
                    fi = xmacello.f
                    Dim dsi As Date = CDate(si)
                    Dim dfi As Date = CDate(fi)
If Not (dSS >= dsi And (dSS.AddMinutes(tempodicarico)) <= dfi)
Then  risposta.Text = "finestra temporale errata: macello:" &
xmacello.codice
                        Exit Sub
                        End If
dSS = dSS.AddMinutes(tempodicarico)
RAATTUALE = dSS

‘ultimo macello
If i = iM - 1 Then
dSS = dSS.AddMinutes(tmaccon)
dSS = dSS.AddMinutes(tk)
ORAATTUALE = dSS
If Not (dSS >= s_conc And dSS <= f_conc) Then
risposta.Text = "finestra temporale errata: conceria"
                            Exit Sub
                        End If
                    End If
macelloprecedente = x.SelectedItem
        Next

'carico max camion
Dim caricomax As Decimal = 0
caricomax = Mezzo.CaricaMezzo(listaMezzi.SelectedItem).qk

If caricoattuale > caricomax Then
risposta.Text = "peso max superato"
            Exit Sub
            End If
```

Verifica che le finestre temporali imposte dalla pelle fresca per lo scarico in conceria siano rispettate, calcolando contemporaneamente anche il costo totale:

```
For i = 0 To lme.Count - 1
namecombo = lcombo(lme.Keys(i))
x = CType(Controls(namecombo), ComboBox)
If x.SelectedItem <> "" Then

If i = 0 Then  ‘primo macello
'tempo max pelli
```

```
Dim macmez As GestioneDati.GestioneDati.MacelloMezzo.Dati =
GestioneDati.GestioneDati.MacelloMezzo.CaricaMacello_Mezzo(x.Sel
ectedItem, listaMezzi.SelectedItem)
                tmik = macmez.Tdepmacello
                tempodicarico = macmez.TempoCarico
                Dim xmacello1 As Macello
                xmacello1 = Macello.CaricaMacello(x.SelectedItem)
                tmax = xmacello1.gi

      Dim tm As TimeSpan = CDate(tmax).Subtract(CDate("00:00"))
                 dSS1 = CDate(ss)
                 Dim dRR As TimeSpan
                 dSS1 = dSS1.AddMinutes(tmik)
                 dSS1 = dSS1.AddMinutes(tempodicarico)

                 dRR = ORAATTUALE.Subtract(dSS1)

If dRR.TotalMinutes > tm.TotalMinutes Then
risposta.Text = "Tempo di permanenza delle pelli provenienti dal
macello " & xmacello1.codice & " superiore a quell massimo"
                        Exit Sub
                    End If
'costo
Dim c_carico As Double = CDbl(macmez.Costodicarico)
Dim c_depmac As Double = CDbl(macmez.Cdepmacello)
costo_tot = costo_tot + CDbl(ck) + c_carico + c_depmac
End If

If i > 0 And i < iM Then
'tempo max pelli
Dim macmacmez As
GestioneDati.GestioneDati.MacelloMAcelloMezzo.Dati =
GestioneDati.GestioneDati.MacelloMAcelloMezzo.CaricaMacello_Mace
llo_Mezzo(macelloprecedente, x.SelectedItem,
listaMezzi.SelectedItem)
Dim tempodiviaggio As String = macmacmez.Tempo
dSS1 = dSS1.AddMinutes(tempodiviaggio)
Dim xmacello As Macello
xmacello = Macello.CaricaMacello(x.SelectedItem)
tmax = xmacello.gi
Dim tm As TimeSpan = CDate(tmax).Subtract(CDate("00:00"))

Dim macmez As GestioneDati.GestioneDati.MacelloMezzo.Dati =
GestioneDati.GestioneDati.MacelloMezzo.CaricaMacello_Mezzo(x.Sel
ectedItem, listaMezzi.SelectedItem)
                    tempodicarico = macmez.TempoCarico
                    dSS1 = dSS1.AddMinutes(tempodicarico)
```

```
                    Dim dRR As TimeSpan
                    dRR = ORAATTUALE.Subtract(dSS1)

If dRR.TotalMinutes > tm.TotalMinutes Then
risposta.Text = "Tempo di permanenza delle pelli provenienti dal
macello " & xmacello.codice & " superiore a quell massimo"
                        Exit Sub
                        End If
'costo
Dim c_macmac As Double = CDbl(macmacmez.Costo)
Dim c_carico As Double = CDbl(macmez.Costodicarico)
costo_tot = costo_tot + c_macmac + c_carico


If i = iM - 1 Then ‘ultimo macello
‘costo
Dim c_macconc As Double = CDbl(macmez.Cmacelloconc)
costo_tot = costo_tot + c_macconc + c_scarconc
                    End If
‘tempo maxpelli
Dim t_tot As TimeSpan = ORAATTUALE.Subtract(CDate(ss))

costo_tot = costo_tot + (t_tot.TotalMinutes * CDbl(ctk) / 60) +
(dRit.TotalMinutes * cdk / 60)
            End If
            End If

macelloprecedente = x.SelectedItem
        Next
risposta.Text = "Soluzione ammissibile; costo= " &
Math.Round(costo_tot, 2)
```

CONCLUSIONI

Nella presente trattazione è stato affrontato il problema del CVRPTW tramite l'implementazione di un programma in Visual Balic che, dato un mezzo ed un percorso definito dall'utente, ne verifica la fattibilità coerentemente ai vincoli di capacità e delle finestre temporali e in caso di fattibilità positiva ne calcola il costo totale.

Questo programma non fornisce la soluzione al problema del CVRPTW, ma costituisce certo un punto di partenza per l'implementazione del programma risolutivo vero e proprio.

Tale programma, utilizzando tutte le informazioni relative a mezzi, depositi, macelli e conceria inserite nel database, dovrebbe selezionare, tramite una serie di permutazioni, tra tutti i percorsi possibili nel rispetto dei vincoli quello a cui corrisponde il minor costo totale che rappresenta quindi la soluzione ottimale esatta del CVRPTW.

Il lavoro svolto in questa tesi si è quindi limitato all'analisi del percorso svolto da un mezzo che parte dal proprio deposito, visita un certo numero di macelli e scarica presso la conceria.

Esso svolge cioè la parte concreta di verifica e calcolo che dovrà essere iterata per ogni combinazione mezzo-macelli-conceria al fine di individuare la soluzione migliore.

BIBLIOGRAFIA

Morito Tsutsumi, Kiwamu Kato, *Applyng Vehicle Routing Problem with time windows to day care courtesy bus service.* Institute of Policy and Planning Sciences, University of Tsukuba. Journal of the Eastern Asia Society for Transportation Studies, Vol.5, October, 2003 (375-385)

Xiangpei Hu and Minfang Huang, *An intelligent solution system for a vehicle routing problem in urban distribution*. Institute of Systems Engineering, Dalian University of Technology. Volume 3, Number 1, February 2007 (189—198)

Herminia I. Calvete, Carmen Galé, María José Oliveros and Belén Sánchez-Valverde, *Vehicle Routing Problems with soft time windows: an optimization based approach.* Monografías del Seminario Matemático García de Galdeano 31, 2004 (295–304)

Liong choong Yeun, Wan Rosmanira Ismail, Khairuddin Omar & Mourad Zirour, *Vehicle Routing Problem: models and solutions*. Journal of Quality Measurement and Analysis, 4(1) 2008, (205-218)

N. Christofides, A. Mingozzi, P. Toth, *Exact Algorithms for the Vehicle Routing Problem, based on spanning tree and shortest path relaxations*. Mathematical Programming 20, 1981 (255-282)

Massimo Paolucci, *Problemi decisionali nei trasporti: Vehicle Routing Problems*, D.I.S.T. Università di Genova

Marco Pranzo, *Appunti sul Vehicle Routing Problem*. Corso di Ottimizzazione su Reti, 2007/2008

Daniele Vigo, *Il Vehicle Routing Problem: Modelli ed Algoritmi.* D.E.I.S. Università di Bologna

Olli Bräysy, Michel Gendreau, *Vehicle Routing Problem with Time Windows, Part I: Route Construction and Local Search Algorithms.* Transportation Science Vol. 39, No. 1, February 2005 (104–118)

Olli Bräysy, Efficient Local Search Algorithms for the Vehicle Routing Problem with Time Windows. MIC'2001 - 4th Metaheuristics International Conference, July 16-20, 2001 (299-303)

Gilbert Laporte, *The Vehicle Routing Problem: An overview of exact and approximate algorithms*. European Journal of Operational Research 59, 1992 (345-358)

Amar Rachman, Arian Dhini and Najuwa Mustafa, *Vehicle Routing Problems with differential evolution, algorithm to minimize cost*. The 20th National Conference of Australian Society for Operations Research & the 5th International Intelligent Logistics System Conference, (78.1-78.13)

Ester Stegers, *A Solution Method for Vehicle Routing Problems with Time-Dependent Travel Times*. Delft University of Technology December 11, 2009

Srisawat Supsomboon*, A Mathematical Model for Vehicle Routing of Used-oil Collection in Bio-diesel Production Using Visual Basic Interface*: *Village Bank and Bio-diesel Project.* KKU Engineering Journal Vol.37, No.2, April - June 2010 (151 - 159)